VisRepNet: 深度学习网络的图解式可视化方法

(申请清华大学工学硕士学位论文)

培养单位：全球创新学院

学　　科：数据科学与信息技术

研 究 生：柯　兰

指导教师：唐　杰　教　授

二〇二一年四月

Visually Representing Networks (VisRepNet): A Diagrammatic Visualization for Describing Deep Neural Networks

Thesis Submitted to

Tsinghua University

in partial fulfillment of the requirement

for the degree of

Master of Science

in

Data Science and Information Technology

by

Joshua Clancy

Thesis Supervisor: Professor Jie Tang

April, 2021

学位论文指导小组、公开评阅人和答辩委员会名单

指导小组名单

唐杰	教授	清华大学

公开评阅人名单

史元春	教授	清华大学
吕勇强	副教授	清华大学

答辩委员会名单

主席	饶培伦	教授	清华大学
委员	唐杰	教授	清华大学
	赵虹	副教授	清华大学
	杨铮	副教授	清华大学
	吕勇强	副研究员	清华大学
秘书	张倩	助理研究员	清华大学

关于学位论文使用授权的说明

本人完全了解清华大学有关保留、使用学位论文的规定，即：

清华大学拥有在著作权法规定范围内学位论文的使用权，其中包括：（1）已获学位的研究生必须按学校规定提交学位论文，学校可以采用影印、缩印或其他复制手段保存研究生上交的学位论文；（2）为教学和科研目的，学校可以将公开的学位论文作为资料在图书馆、资料室等场所供校内师生阅读，或在校园网上供校内师生浏览部分内容；（3）按照上级教育主管部门督导、抽查等要求，报送相应的学位论文。

本人保证遵守上述规定。

（保密的论文在解密后遵守此规定）

作者签名：　　　　　　　　导师签名：Jie Tang

日　　期：2021/04/07　　　日　　期：April 8, 2021

摘　要

本文提出了视觉表示网络（Visually Representing Networks，VisRepNet），一种用于描述深度神经网络的图解式可视化方法。VisRepNet 的目标是帮助用户学习，理解和交流深度神经网络。本文选取了常见的，不同架构和复杂度的深度神经网络，并对这些网络里涉及的概念和组件进行整理、总结与归类。之后，本文基于当前对图表设计、符号学和认知语言学的研究，创建了 VisRepNet 的描述图形设计准则。本文进而进行了用户需求实验，研究用户对深度神经网络可视化的需求与兴趣维度。根据用户的需求与设计准则，本文将 VisRepNet 设计为一种涵盖多个过程抽象级别的层级结构描述方法。在参考了用户自行绘制的图示以及所提出的设计建议之后，本文创建了一套表示深度神经网络中组件的图示符号—VisRepNet。用 VisRepNet 描述的深度神经网络，既可以提供网络的概要，也可以展示其函数和参数的底层细节。本文通过定量与定性的用户研究评估了 VisRepNet 的有效性。在定量研究中，本文发现 VisRepNet 帮助学习的能力明显胜于现有的其它可视化方法（U=80.5, Z=-2.28, p=0.02。在定性研究中，本文发现 VisRepNet 的用户在具体性、完整性、理解力、对编程和学习的作用性等方面的表现均显著优于对照组。

关键词：深度神经网络，深度神经网络可视化，图表设计，神经网络架构可视化

Abstract

In this paper, we present Visually Representing Networks (VisRepNet), a diagrammatic visualization for describing deep neural networks. VisRepNet's goal is to help users learn, understand, and communicate deep neural networks. We started by categorizing concepts and components from a set of popular deep neural networks with a variety of architectures and complexities. Following this, we collected a compilation of research into diagram design, semiotics, and cognitive linguistics to create diagram design guidelines. We then conducted user studies to investigate user's needs and dimensions of interest for deep neural network visualizations. Based on user's needs and design guidelines, we designed a structure of multiple procedural abstraction levels for VisRepNet. Then, including design recommendations from a user-defined drawing study, we created symbols to represent components in deep neural networks. Therefore, VisRepNet can describe a deep neural network by providing not only a high-level overview of the network's architecture and update environment, but also a low-level detailed view of its functions and hyper-parameters. We evaluated VisRepNet's effectiveness through a quantitative and qualitative user study. In the quantitative study, we found that VisRepNet significantly outperforms current visualization methods regarding the ability to help with learning ($U = 80.5, Z = -2.28, p = 0.02$). In the qualitative study, we found that users perceived performance of VisRepNet was significantly higher than our control in terms of specificity, completeness, ease of understanding, usefulness for coding, and usefulness for learning.

Keywords: Deep Neural Network; Deep Neural Network Visualization; Diagram Design; Neural Network Architecture Visualization

Contents

List of Figures and Tables

Chapter 1 Introduction

The field of deep learning is progressing rapidly in many directions. Deep learning researchers often present and communicate deep neural networks using visualization tools such as NN-SVG[1] and PlotNeuralNet[2]. Related works shows that visualizations can have a significant effect on people's understanding[3-4]. Therefore, designing an effective and intuitive deep neural network visualization method is important for the deep learning community. However, as our user studies show, many visualizations of neural architecture are considered inadequate. This seems to be because either the visualizations does not provide enough detail to recreate the network, or the visualization does show adequate detail but becomes inaccessible to users in the process.

In this paper, we present Visually Representing Networks (VisRepNet), a diagrammatic visualization for describing deep neural network architectures and their related update environments. VisRepNet is different from existing visualization tools that only represent a simplified summary of network architectures[1-2,5-11] , or a large detailed diagram of network architectures,[12-29] or specialize towards visualizing specific features [29-35]. VisRepNet represents deep neural networks by providing both a high-level summary view of the network architecture and also a low-level detailed view of functions and hyper-parameters. On top of this, VisRepNet also includes features that few prior works exhibit, namely, VisRepNet displays the training environment of neural networks, displays both tensors and functions, and uses discrete symbols. These various features were implemented with user needs primarily in mind. VisRepNet is designed to allows users to more easily access and understand the context, system, and premise of a deep neural network.

To design VisRepNet, we first summarized and categorized the concepts and components within deep neural networks. We did this by exploring various popular neural networks with a wide range of architectures and training environments. During this process we also identified common metaphors that are used in teaching deep learning concepts. This resulted in a list of components and a list of metaphors that describe these components. On top of this we searched for generalizations in deep learning concepts which we could exploit in our visualization. For example, convolutions can be represented in a general form. So that not only spatial convolutions are represented, but also convolutions that convolve over channel, group, and temporal dimensions.

Second, we created diagram design guidelines from research into diagram design, Semiotics, and Cognitive Linguistics. These guidelines outline heuristics that guided our design decisions. For example, the first guideline is to not create new symbols for a meaning, when there are already well known and established symbols for that meaning. This guideline justified our use of common linear algebra and activation function symbols. Once collected, these design guidelines provided us with a framework that we could continually refer to during our design process.

We also conducted user studies to investigate user's needs and dimensions of interest towards deep neural network visualizations. We were interested in what types of deep neural network visualizations our users liked, and what specific features they would want represented. This was done via a survey with both open ended questions, as well as multiple-answer questions. To summarize, we found that users want to have clear, well-organized diagrams that show both a high-level overview of the network and training environment, as well as a low-level view of functional details including hyper-parameters. On top of this users also want to see the flow and changing of tensor shape, as the tensors interact with functions. It was from this survey that we gathered the basic requirements of VisRepNet.

Based on these results, we then employed a user-defined design methodology to collect user-defined symbols that represent deep neural network components. This was done through a drawing survey, where our users visualized specific deep learning components. For each component, we aggregated user drawn data and collected a set of visual features which our users most-often employed. These common features provided us with a set of design recommendations for how to symbolize each component. This drawing exercise thus grounded our symbolic design in data of how our users represented deep learning concepts.

Using the data from the deep neural network research, diagram design guidelines, user needs survey, and user-defined drawing exercise, we created VisRepNet. VisRepNet is organized via procedural abstraction levels as to separate high-level and low-level visualizations. The higher-level visualizations provide information about a network's training environment, including its input, output, label, loss, and update method. The lower-level visualizations then provides information about a networks architecture, including the various functions and hyperparameters involved. VisRepNet then further includes an area for the diagram author to elaborate on their specific focus. This is so, that in academic papers, VisRepNet can provide the context needed to understand the paper's focus and area of improvement.

Within this paper, we go step by step through each of the design decisions made to create VisRepNet and we justify the reasoning behind these decisions. All justifications come from the proceeding studies of deep neural network components, diagram design guidelines, user needs, and user-defined symbols.

Finally, we evaluated VisRepNet's performance and effectiveness through a quantitative and qualitative user evaluation study. In the quantitative study, we found that VisRepNet significantly outperforms current visualization methods regarding the ability to help with learning ($U = 80.5, Z = -2.28, p = 0.02$). In the qualitative study, we found that users perceived performance of VisRepNet was significantly higher than our control in terms of specificity, completeness, ease of understanding, usefulness for coding, and usefulness for learning.

This paper's contributions are threefold:

1. We investigated user's needs and dimensions of interest for deep neural network visualizations through a user survey study.
2. We created VisRepNet, a diagrammatic visualization to describe deep neural networks. VisRepNet's design is guided and justified by user needs, user-defined symbolic designs, deep learning research and diagram design guidelines.
3. We evaluated VisRepNet's effectiveness through quantitative and qualitative user studies. Results show that VisRepNet significantly outperforms current deep neural network visualization methods regarding its ability to help with learning and user perceived performance.

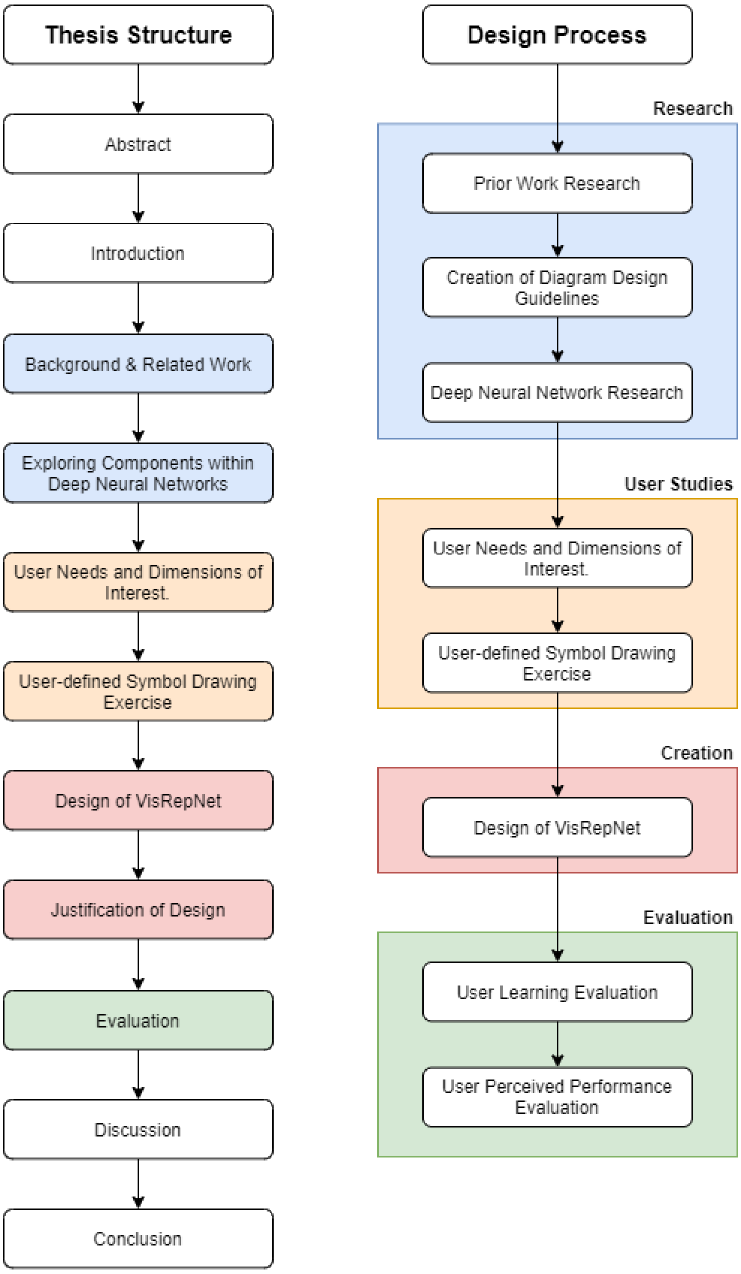

Thesis Structure
Abstract
Introduction
Background & Related Work
Exploring Components within Deep Neural Networks
User Needs and Dimensions of Interest.
User-defined Symbol Drawing Exercise
Design of VisRepNet
Justification of Design
Evaluation
Discussion
Conclusion
Design Process
Research
Prior Work Research
Creation of Diagram Design Guidelines
Deep Neural Network Research
User Studies
User Needs and Dimensions of Interest.
User-defined Symbol Drawing Exercise
Creation
Design of VisRepNet
Evaluation
User Learning Evaluation
User Perceived Performance Evaluation

Figure 1.1 Architecture of this Thesis

Chapter 2 Background and Related Work

In this section, we summarize related work in deep neural network visualizations and user centered design. We also summarize the research that allowed us to derive our diagram design guidelines.

2.1 Deep Neural Network Visualizations

Visualizations are very important within the field of deep learning. They provide a means to explore opaque concepts, consider high dimensional data, and understand complex architectures. Visualizations help beginners learn, practitioners work, and researchers understand[36-39].

Visualizing the gradients of neural networks is a common method[18,30-35,40-45]. This technique is useful, as it can help practitioners discover and understand issues during training. Another common method is to visualize the input features that cause individual neurons to produce output[11,28,46-54]. The goal of these visualizations is to see within the ”black box”. Such methods allow for a greater understanding of how and why trained neural networks produce their output. Another visualization technique projects high-dimensional data into two or three dimensions as to be visualized[55-60]. This method allows practitioners to better interpret this data. Further to the aforementioned methods, there is a niche of interactive visualizations, that have been created for educational purposes[33,61]. These allow beginners to better understand the components and concepts, that comprise deep learning.

The visualization of a neural network’s architecture is also a common goal. Some of these architectural visualizations are specific to a certain type of network, or even a specific part of a network. Examples of this are visualizations that specifically represent convolutional neural networks[1,17], or recurrent neural networks[62-65], or transformer attention mechanisms[66-68]. Other visualizations attempt to represent a wide array of architectures. This is generally done by representing the flow of data through functional components, with the names of these components and their attributes being taken directly from the code of the neural network itself.[1-2,5-29]

There is a large variety in how these projects visualize neural network architectures. Some display blocks of text to represent components, while others display visual symbols.

Some create sprawling graphs that need to be carefully navigated, while others create compact visualizations that lack specific details. Some visualizations specifically represent the functions within a network's architecture, while others specifically represent the tensors that flow through the architecture. To the best of our knowledge, the variety between these different visualizations is due to different imagined users. However this is hard to tell, since most of these different projects are not academic in nature. There is thus little public accompanying user research or design justification for these visualizations.

Perhaps most interesting is what these visualizations seem to lack. Few, if any, of the found prior work have the following attributes. Please refer to 2.1 for specifics. First, they do not visualize the training environment of neural networks, instead they focus on the architecture only. We assume this choice has been made because certain architectures can be used in multiple training environments. Whilst this is true, we will argue that the training environment is often the most important feature to understand when considering the functioning of a neural network and should be visualized. Second, few works visualize the architecture at multiple levels of detail. Thus most prior works either show a large detailed visualization or a summarised and simplified visualization. Third, few prior works show both tensors and functions, they usually choose only one of these to visualize. This is comparable to visualizing a production process by only showing the products produced in between each step, or only showing the machines that produce those products at each step, not both. Finally few prior works, break their visualization into discrete symbols. Instead prior works generally rely on written text to indicate the differences between components. Discrete symbols offer advantages and disadvantages. Discrete symbols allow for faster recognition and search through a visualization, however they must be taught[3,69-71].

In regards to the purpose of the visualization, the most similar projects to our own work are NN-SVG: Publication Ready Architecture Schematics[1] and PlotNeuralNet[2]. These have created visualization systems for clearly communicating neural network architectures specifically for academic papers. These projects have built automated, fast diagram creation systems. These result in professional, good looking diagrams. However these diagrams ignore many of the requirements that we found within our user studies. For example, these diagrams do not represent the training environment, do not display the architecture at multiple levels of detail, cannot represent certain architectures and functions, and do not represent many hyperparameters.

In regards to level of detail, the most similar project to our work is Tensorboard[18].

Table 2.1 Prior work on Visualizing Deep Neural Networks

Architecture Visualization	Automatic visualization	Displays Neural Functions	Displays all other functions	Displays hyperparameters	Displays Tensors	Uses Discrete Visual Symbols	Uses multiple levels of detail	Displays Training Environment	Clear and concise on paper
VisRepNet		✓	✓	✓	✓	✓	✓	✓	✓
NN-SVG[1]	✓	✓			✓				✓
PlotNeuralNet[2]	✓				✓				✓
TensorBoard[18]	✓	✓	✓	✓			✓		
Caffe[19]	✓	✓	✓						
MatLab[20]	✓	✓	✓			✓			
Netron[23]	✓	✓	✓	✓	✓				
Keras.js[21]	✓	✓	✓	✓					
Keras sequential ascii[22]	✓	✓	✓		✓				
Visual Keras[6]	✓			✓					✓
Net2Vis[5]	✓	✓				✓			
Draw ConvNet[17]	✓	✓		✓					
GraphVis[8]	✓	✓							
ENNUI[9]		✓	✓			✓			
TensorSpace[11]	✓				✓				
Netscope[12]		✓	✓	✓					
Moniel[13]		✓	✓				✓		

Tensorboard displays a neural network's architecture with a high amount of detail, and even includes some information about a neural network's training environment. However, tensorboard's diagrams are not formatted to be clearly and concisely displayed on paper for communication purposes. Instead, these diagrams seem to have been designed to display as much information as possible, on a computer, for the creators of the neural network. Despite its detail and size Tensorboard diagrams are still missing elements that our users requested. For example, Tensorboard does not visually show the changing shape of tensors.

2.2 User Centered Design

User Centered Design is a framework to keep oneself orientated whilst designing[72-76]. In this framework, a motivation and goal should first be established around a user group. This goal can grow organically, but its definition should always be at the center of attention. Then, a set of methods are available to gather information from that user group about their wants and needs regarding the goal. From here, an iterative design process begins, where prototypes are made, and users are consulted on the prototype design. At each step one should ground design decisions via user collected data. In this way the design is not the product of one designer, but instead the product is designed by the interested user group, as facilitated by the designer.

When it comes to constructed diagrammatic visualizations, the vast majority were created before User Centered Design methods became common practice. An example of a recently constructed diagrammatic visualization for science would be Feynman Diagrams[77]. This diagram form was not created by user centered design, but instead via the imaginings of one individual. Regardless, these visualizations are particularly well suited for quantum physics and represent a major achievement. Two possible examples of a diagrammatic visualization that were designed by methods approaching User Centered Design are IConji[78] and Emojis. However neither of these projects documented their work for academic review.

We took particular inspiration from the methodology of Human Computer Interaction papers. Here, there is a set of projects that attempt to find sets of gestures, that users would find intuitive[79-83]. These projects incorporate user testing where participants are asked to communicate an idea via gesture. With this data, these studies attempt to find intuitive commonalities, resulting in a language of gestures for a given system. We do something similar here. We have users communicate deep learning concepts through a symbol drawing exercise.

2.3 Diagram Design Guidelines

Our justifications for design decisions do not arise entirely from user studies. We also use distilled diagram design guidelines that we have gathered from papers within diagram design, semiotics, and cognitive linguistics. This work is summarized here.

First, when designing diagrammatic visualizations, it is important to know the value of diagrams. First a diagram has value in the abstraction of complex concepts into simplified symbols[3,69]. Such a practice clarifies and indicates the boundaries of those complex concepts. Second, a diagram arranges symbols into clear and informative relationships[84-86]. Such intelligent arrangements reduce the effort of search and provide the user with pathways for their attention to follow. If done well, this will decrease cognitive load[70]. Third, through the illustration of the entire system, diagrams highlight unknown areas of interest and show how those unknown areas interact with the known. Thus a user's attention is provided with context in order to understand concepts and relations that they were not originally aware of. Finally, diagrams can frame symbolic representations within appropriate conceptual metaphors[87-90]. This can help users make the appropriate conceptual connections in order to understand underlying concepts[91-93].

We want our visual diagram to be intuitive and easy to navigate. We thus came up with a set of guidelines to guide our design decisions. These are the following. To quickly learn new symbols and grammar, it is beneficial if they are already in use within current communicative systems[94]. Thus we should use well known symbols if possible. Sometimes however we are considering a meaning for which there is no obvious symbol already in existence. In such a case, we have three main strategies. First, we can use well known symbols as smaller parts within our ultimate symbol. Second, we can use cognitive metaphors that allow people to best understand the meaning. Third, we can visually indicate when a symbol is a part of a group of symbols[71]. These methods together can create intuitive symbols but also have the tendency to create complex ones. Thus we must temper these strategies, with the recognition that symbol simplicity is required for quick recognition and ease of search through the diagram proper. To summarise, our diagram design guidelines are:

1. Do not create new symbols for a meaning, when there are already well known and established symbols for that meaning.
2. Consider using recognizable symbols as smaller parts within your overall symbol.
3. Symbols and grammar should take advantage of cognitive metaphors when possible and useful.
4. Symbols within a grouping should be recognizable as within that grouping.
5. We should be able to quickly recognize symbols. Thus, symbols cannot be too complex.

Chapter 3 Exploring Components within Deep Neural Networks

In this section, we describe our investigation of the concepts and components within deep neural networks. First, we focused on categorizing the components related to deep neural networks. Second, we focused on the communication methods via which these components were taught. Third we focused on finding conceptual generalities that we could exploit in our visualization.

We researched various popular deep neural networks including ResNet[95], Inception[96], UNet[97], DC-GAN[98], SimCLR[99], BERT[100] and Axial Attention Networks[101]. These networks were chosen to include a wide range of different modern network architectures (two categorical encoders, an auto-encoder, an adversarial learning setup, a contrastive learning setup, an NLP attention-based network, and a visual attention-based network.)

3.1 Categorizing Components

We broke down the components within deep neural networks in multiple ways. First we considered whether a component lies within the network's update environment or within the network's architecture.

- **Within the Update Environment:** Components that facilitate the network's learning or processing from outside the network itself.
- **Within the Network Architecture:** Components that help process the network input into the network output.

We then categorized components and concepts via whether they could be described as a mathematical object or as functions.

- **Mathematical Objects:** Components that hold data.
- **Functions:** Components that input, process, and output mathematical objects.

We then created the following descriptive groupings of components. Note, that some components may fall into multiple descriptive groupings.

- **Neural layers** Examples include, linear layers, convolutional layers, and recurrent layers.
- **Activation Functions** Examples include relu, sigmoid, and tanh.

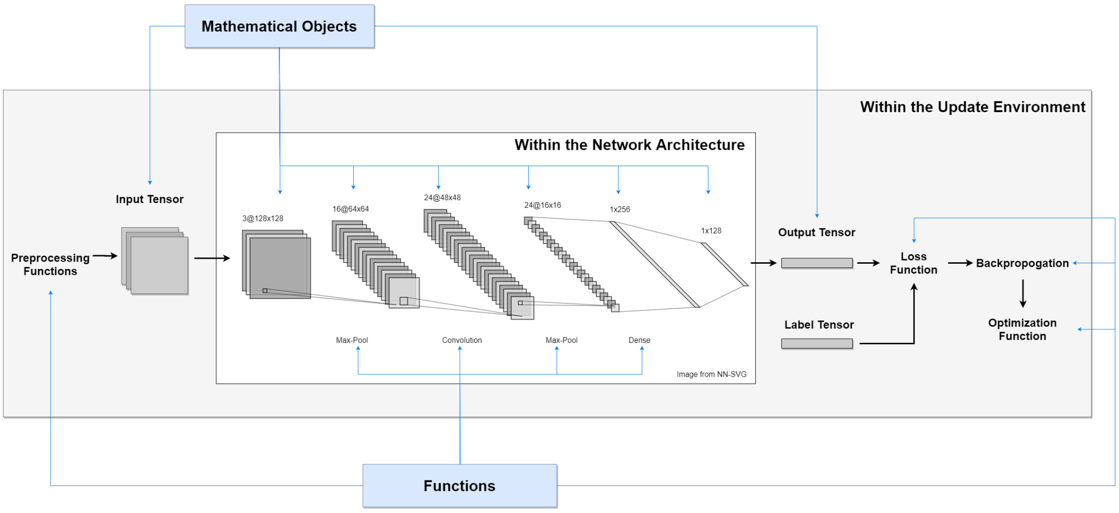


Figure 3.1 Within the Network Architecture vs. Within the Update environment - Architecture is visualized using an NN-SVG diagram[1]

- **Pooling Functions** Examples include upsample, downsample, and max-pool.
- **Regularization Functions** Examples include L1 norm, batchNorm, and drop out.
- **Linear Algebra Functions** Examples include the dot product, matrix multiplication, concat, and reshape.
- **Preprocessing Functions** Examples include rotate, color hue, and standardization.
- **Measurement Functions** These include both loss and reward functions. Examples include MSE, BCE, and KL-divergence.
- **Update Functions** These include both functions that compute the gradient, such as backpropogation, and optimization functions which step along this gradient, such as gradient decent and ADAM.
- **Tensors Objects** Examples include the input, the output, and the label.
- **Internal Objects** This grouping includes the mathematical objects that are within the various functions. Examples include the weight tensors, the bias, various distributions, etc.

3.2 Communication of Components

We also payed close attention to how people communicated and taught deep neural network concepts. We took particular note of any conceptual metaphors that we noticed. Examples of these include:

- The progression of an input tensor to output tensor is often referred to as the input *moving* or *flowing* through functions towards the output (as if along a path).

- The progression of an input tensor to output tensor is also sometimes referred to as the input transforming (via functions) into the output (as if a metamorphosis).
- Convolutions are often explained via a sliding window metaphor.
- Batch-Norm is often explained via a grouping metaphor around the origin of the graph.
- Dropout is often explained via neurons *dropping out* as if leaving as to be not included within the function.
- Adversarial training environments are metaphorically compared to the relationship between a ”forger” and a ”cop”.
- Auto-encoder training environments are metaphorically compared to communication systems with low bandwidth.

3.2.1 Generalizations in Deep Neural Networks

During this research we also focused on finding generalities that we could exploit in our visualization. We found three main generalities that were useful.

First, the interpretations of tensor dimensions seem to always fall within one of the following general categories: channel, spacial, temporal, or grouping. One could describe all tensors that we encountered, via combinations of these categories.

Second, there is a general form for convolutions. Convolutions do not only apply in the spatial dimension but also along grouping and time dimensions. One could represent all convolutions that we encountered by explicitly including these dimensions within the representation of the convolutional function.

Third, we defined a general framework to define the update environment with the INOMU framework. INOMU stands for Input, Network, Output, Measurement, and Update. At a basic level, we can describe each of these as: Network receives **input**, **network** processes input, network produces **output**, output is **measured** based on some “target” state, and then the network **updates** based on that measurement. We claim that the INOMU organization can fit all learning networks at an abstract level. We do not believe that this is a controversial or a surprising claim, we do however provide more detail and evidence of this framework’s generality in our planned, upcoming, theoretical paper RepLeStruct -Representing Learned Structures.

Chapter 4 Exploring User Needs for Deep Neural Network Visualizations

This section outlines the user survey that we conducted to understand user needs and dimensions of interest in relation to deep neural network visualizations.

4.1 Participants

We recruited 25 participants (5 female, 20 male) with an average age of 25.8 (s.d. = 2.5) from the deep learning community in academic universities or industrial companies (e.g., Microsoft). Participants had an average of 2.4 years (s.d. = 1.1) experience in deep learning. Their self-reported deep learning knowledge score had an average value of 4.0 (s.d. = 1.2) using the 7-point Likert scale.

4.2 User Survey Design and Procedure

We begun our design process by idealizing an imagined prototypical user. The needs of this prototypical user can be considered our hypothesis. Our survey was initially designed around confirming or denying this hypothesis.

Our prototypical user is a part of the wave of academics and professionals that are currently exploring the possibilities of deep learning. Our prototypical user has already been taught the fundamentals and has already built multiple deep learning models. Our prototypical user has framed their understanding of neural network architectures and training environments through coding. But our prototypical user has hit a wall. If they read academic papers, they find it difficult for three reasons. First, the volume and velocity at which deep learning papers are produced is staggering. Second, most academic articles do not sufficiently contextualize their discussions for coders to understand. Third, current papers often do not ground their discussion with sufficient details about system architectures and training environments.

To consider this hypothesis and to explore the needs of our users, we utilized an online questionnaire consisting of both open-ended questions and multiple-answer questions. First, we asked each participant about his/her demographic information and experience in deep learning. Then, we asked three questions: 1) *"How do you use network architecture* *visualizations?"* 2) *"What materials did you use to understand deep neural networks?"* 3) *"When you read deep learning materials (e.g., academic papers), how do you like their explanatory visualizations? What do you like, what do you not like?"*

Then, we asked participants to fill in 13 multiple-answer questions to investigate what users want to see in deep neural network visualization. Given our research into deep learning, we already had a list of important functions and concepts that make up deep neural networks. With this list we asked our users what they would want visualized in a network diagram. An example question is: *"You are studying a neural network architecture through a visualization. There is a fully connected layer represented. You get a quick look at it, what do you want to see? Choose all that apply."* Then a list of possible dimensions of interest is presented to the participant:*"The bias, the number of weights, the general expansion or compression of channels, the exact input and output number of channels, an activation function (if there is one), the normalization function (if there is one), and/or other"*. Within this dimension of interest survey we also asked our users some specific design questions, such as our question on procedural abstraction: *"Do you want common sets of layers and functions to be defined separately, with the main visualization referencing these common sets?"* All of these questions are listed in the appendix.

4.3 Results and Analysis

4.3.1 User needs from open-ended questions

We drew the following main conclusions from our open ended questions.

Users want diagrams that show functional details, including hyper-parameters 7 participants noted this, P1 said *"I prefer it when new or modified layers (e.g., inception, modified convolution etc..) are visualized in detail (e.g., attention layers called out, similarities to simpler layers highlighted etc..). Once the author clearly defines layer types, I prefer a simple high-level architecture diagram (i.e., layer names with relevant details like shape, kernel size where appropriate, activation type, etc..)."* P2 commented, *"I like where [visualizations] show the overall structure of the model... but they often lack to mention the exact parameters, dataset modifications, etc. it makes it too hard to reimplement their findings."*.

Well-organized diagrams that clearly show the network at a top-down level are highly demanded. 7 participants noted this, P3 said *"I like where [visualizations] show the overall structure of the model"* P4 said, *"I like clear structure, with large color block* *to clarify different functions etc. I dislike too many words and messy lines and blocks with unclear hierarchy."*

Users want diagrams that go from input to output and show the transformation of tensors as they progress through that journey. 3 participants noted this. P5 said *"I like diagrams where they show how the shape of deep learning [tensors] change as [they go] through each layer... an input to output visualization."*

We also found evidence to support our prototypical user hypothesis. 10 users indicated that they do not read academic papers, despite the user subgroup having an average of 2.2 years of deep learning experience (s.d. = 1.0). Out of the group of users who do read academic papers, 7 participants indicated frustration over academic papers not providing architectural and training details. For example, one user noted *"Some papers leave out the details of their model (like activations and connections) making it hard to reproduce their model."* When considering visualizations specifically, another user said *"Not all of the papers that propose novel architectures include easily understandable visualizations... If a novel recipe is being proposed, it should be immediately clear to the reader what makes it different from existing works from the visualization diagram."* Yet another user remarked *"few [deep learning papers] have good graphs, most have nothing… it is hard to know the details."*

4.3.2 Dimension of Interests

Table 4.1 shows the data from the multi-answer questions. Here we present the importance of components with a percentage-based agreement metric. For example, eighty percent of participants agreed that they wanted to see batch normalization functions within network visualizations. When the percentage of participants who agreed on a dimension of interest was above 60%, we made that dimension of interest a visualization requirement for our diagrammatic language. Altogether, this gave us a list of requirements that we needed to visualize. This list of requirements thus constrained our design process based on user wants.

Table 4.1 Design Requirements derived from user consensus (% of users who want x visualized)

Dimension of Interest for Visualization	**User consensus(%)**
Use Procedural Abstraction	95%
Visualize the dot product between two tensors	85%
Visualize adding two tensors together	85%
Visualize Concatenating a tensor	85%
Visualize tensor shape of the output	85%
Visualize normalization functions	80%
Visualize the exact input and output dimensions for a given neural function	80%
Visualize slicing a tensor	80%
If applicable, visualize multiple outputs, and where they exit	80%
Left to Right flow of information	75%
Visualize Activation Functions	75%
Visualize the general expansion or compression of channel dimension.	75%
Visualize kernel information for a convolutional layer	75%
Visualize matrix multiplication between two tensors	75%
Visualize Up and Down sampling	75%
Visualize an output example	75%
Visualize the loss function used, including its specific equation	75%
Visualize tensor shape between functions	70%
Visualize an Input example	70%
Visualize Reshaping of a tensor	70%
Include information about the general preprocessing functions	65%
Include input shape information	65%
Visualize the type of output	65%
Visualize an example of the label	65%
If applicable, visualize multiple inputs, and where they enter	60%
Visualize Step size for a convolutional layer	60%

Chapter 5 User-defined Symbolic Design for Deep Neural Network Components

5.1 Participants

In total we had 15 participants for this study (4 female, 11 male) with an average age of 25.3 years (s.d. 2.24). This group consisted of academic students and industry professionals. Participants had an average of 2.5 years (s.d. 1.2) of deep learning experience.

5.2 User-defined Drawing Exercise Design and Procedure

Researchers in the Human-Computer Interaction (HCI) community have adopted a user-defined methodology to design gesture-interfaces for a variety of tasks[79-83]. These researchers obtain an intuitive gesture set, for a set of commands, by summarizing the common gestures among a group of users. This user-defined design method has proved to be effective for this purpose. In this study, we adapted this user-defined design method towards deriving symbolic representations of components in deep neural networks.

However, the domain of gesture design is different to the domain of visual symbolic design. There are simply far more dimensions of possibility when considering visual symbolic design. Thus, it is not feasible to simply group the visual symbols based on similarity and use the most commonly employed one. Instead, we used a different strategy.

We created categories of features employed within user symbols. We isolated some aspect and created categories to describe the user drawn symbols in relation to that aspect. For example, was the user symbol 2d or 3d? We have an aspect of interest (The dimensionality of the symbol) and we have two categories to describe the user drawings in relation to that aspect of interest (2d or 3d). One of the most important of these classification schemes was: ”did the user represent the function alone, the mathematical objects that the function acted upon, or both?” This was important because for many deep learning functions, it was common for users to visualize the function, by representing the input and output tensors, not the function itself.

We also came up with specific categorization schemes for individual components. This was necessary as each component has different aspects of interest and different categories within those aspects of interest that describe the population of user symbols. For example, the SoftMax function was symbolized in many ways. The Softmax *function symbol shape* aspect had these four categories.

- User symbol for Softmax function uses a box with a label. (36%)
- User symbol for Softmax function uses a ”S” shaped curve. (29%)
- User symbol for Softmax function indicates one value increasing, while all others decrease (21%)
- User symbol for Softmax function does something else. (other) (14%)

For each symbol then, we had aspects of interest, within which were related categories pertaining to possible methods of representation. This allowed us to break down symbols in a manner that we could objectively consider. Since for each aspect, we could now calculate which method was most often employed by our users. The most used methods we then turned into design recommendations for our own visual symbolic designs.

5.2.1 Categorical Agreement

Given our users symbols we used a category-based agreement metric to calculate consensus among our users. This agreement metric goes beyond simply calculating which category is most used, and includes information about the number of categories. (This process has been used in related works[83,102])

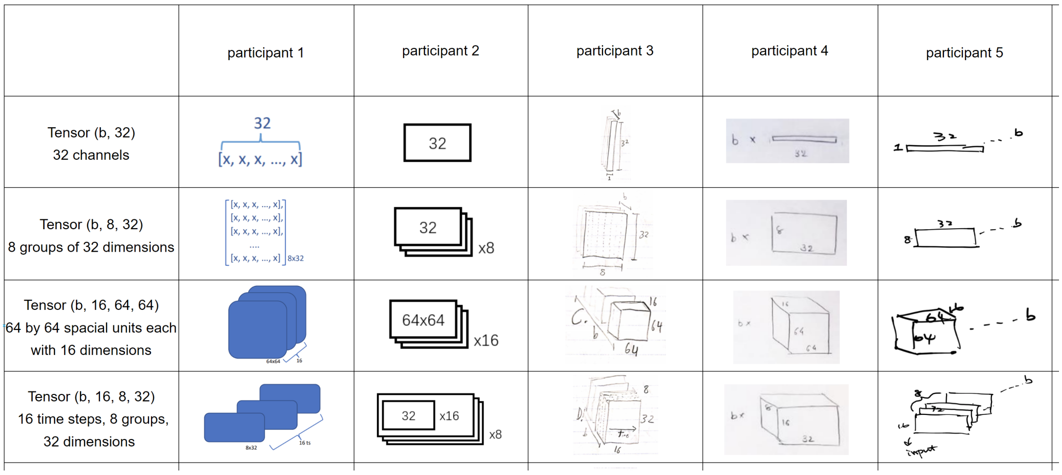
$$A = \frac{\sum_{r \in R} \sum_{P_i \subseteq P_r} \left(\frac{|P_i|}{|P_r|} \right)^2}{|R|} \tag{5-1}$$

In Equation 1, r is a referent (a meaning that is referenced via user symbols), and R is the set of referents. P_r is the set of symbols that refers to r, and P_i is a subset of those symbols. Our categories then, describe different methods of creating P_i subsets.

For each question, we used the categorical agreement metric to calculate user agreement in the different categorical schemes. If a user drawing was unclassified in a given scheme it was considered ”other”. If considered ”other” we treated that drawing to have its own separate categorization. Thus the agreement metric takes into account those user drawings which defied our classification schemes.

Note that we used a simple percentage agreement in 5.1. The categorical agreement metric was used separately to weight user recommendations. If a categorical scheme has low total categorical agreement (<0.5), then we weight the user recommendations accordingly. Thus we have strong recommendations and weak recommendations.

Figure 5.1 Examples of user created representations of deep learning concepts.



5.3 Results

The results are shown in the following tables. Due to the multi-categorical nature of the data, it is difficult to show everything. Therefore in table 5.1 we show the design decisions that we found most valuable and the percentage of users who employed those decisions. And in table 5.2 we show the categorical agreement for each set of categories that we used. On top of this, we also show in this table the number of groupings for each aspect, and how many user drawings fell into an ”other” category if applicable.

Table 5.1 Top Design Recommendations derived from user drawn symbols.

Component	**Design Recommendation 1**	% used	**Design Recommendation 2**	% used	**Design Recommendation 3**	% used
(b, 32c) tensor	2d tensor symbol = a rectangle	80%	Did not visualize individual units	67%	visualized batch dimension	40%
(b, 8g, 32c) tensor	3d tensor symbol = stack of rectangles	67%	Visualized 8 32*1 rectangles	73%	Did not visualize individual units	60%
(b, 16c, 64x, 64y) tensor	3d tensor symbol = 3d shape	47%	Visualized x*y spatial dim with z axis = channels	80%	Did not visualize individual units	80%
(b, 16t, 8g, 32c) tensor	3d tensor symbol = stack of rectangles	53%	No differentiation of time	80%	Did not visualize individual units	86%
Linear layer (16c to 32c)	User visualizes both tensors and function	73%	function symbol = Rhombus shape and/or arrow	60%	Function symbol is 2d	93%

Linear layer (32c to 16c)	User visualizes both tensors and function	73%	function symbol = Rhombus shape and/or arrow	60%	Function symbol is 2d	93%
Linear layer (16c to 16c)	User visualizes both tensors and function	73%	function symbol = Rhombus shape and/or arrow	60%	Function symbol is 2d	93%
(1x, 1y) kernel	Kernel symbol = Square with units	87%	symbol is 2d	87%	User visualized math object (kernel) only	87%
(3x, 3y) kernel	Kernel symbol = Square with units	87%	symbol is 2d	87%	User visualized math object (kernel) only	87%
(5x, 5y) kernel	Kernel symbol = Square with units	87%	symbol is 2d	87%	User visualized math object (kernel) only	87%
Convolutional layer (3,3) kernel (2) step	User visualizes both tensors and function	73%	function symbol = arrow connecting kernels	60%	horizontal flow of information	87%
Convolutional Transpose layer (3,3) kernel (2) step	User visualizes both tensors and function	73%	function symbol = arrow connecting kernels	60%	horizontal flow of information	87%
Convolutional layer (1,1) kernel (1) step	User visualizes both tensors and function	73%	function symbol = arrow connecting kernels	60%	horizontal flow of information	82%
Up Sample	User visualizes both tensors and function	86%	function symbol = arrow connecting tensors (small to big)	71%	horizontal flow of information	93%
Down Sample	User visualizes both tensors and function	86%	function symbol = arrow connecting tensors (big to small)	71%	horizontal flow of information	93%
Reshape Up	User visualizes both tensors and function	86%	function symbol = arrow connecting tensors	67%	horizontal flow of information	93%
Reshape Down	User visualizes both tensors and function	86%	function symbol = arrow connecting tensors	67%	horizontal flow of information	93%

Drop out	symbol = Crossed out/Blacked out neurons	40%	2d Design	100%	User visualizes function only	100%
Batch norm	function symbol = box with label	46%	function symbol = indicates squashing of values	38%	User visualizes function only	92%
Soft Max	function symbol = S shaped curve	29%	function symbol = soft max effect on values	21%	2d Design	100%
Abstract representation of Loss	symbol = output compared to label	45%	symbol = ”L”	36%	Visualize both functions and tensors	62%
Abstract representation of Update	symbol = arrow/s looping	50%	symbol = system diagram	25%	symbol is 2d	100%

Table 5.2 Categorical Agreement for Categories.

Component	**Aspect**	**# of groups**	**# in ”other”**	**Categorical Agreement**
Tensor	location of dimensions?	2	NA	0.68
Tensor	2d or 3d representation?	2	NA	0.768
Tensor	symbol shape?	3	4	0.3877
Tensor	visualize individual units?	2	NA	0.68
Tensor	visualize batch dimension?	3	NA	0.395
Linear Layer	visualize function, tensors, or both?	3	NA	0.608
Linear Layer	function symbol is 2d or 3d?	2	NA	0.875
Linear Layer	function symbol shape?	3	4	0.395
Linear layer	horizontal or vertical information flow?	2	NA	0.555
Kernel	2d or 3d representation?	2	NA	0.768
Kernel	symbol shape?	2	2	0.76
Kernel	see tensor under kernel?	2	NA	0.503
Convolution	Conv function symbol shape?	3	4	0.395
Convolution	visualize function, tensors, or both?	3	NA	0.608
Convolution	function symbol is 2d or 3d?	2	NA	0.555
Convolution	horizontal or vertical information flow?	2	NA	0.768

Up/down sample	visualize function, tensors, or both?	3	NA	0.755
Up/down sample	Up/down sample function symbol shape?	3	2	0.541
Up/down sample	function symbol is 2d or 3d?	2	NA	1
Up/down sample	horizontal or vertical information flow?	2	NA	0.867
Reshape	visualize function, tensors, or both?	3	NA	0.755
Reshape	Reshape function symbol shape?	2	5	0.466
Reshape	function symbol is 2d or 3d?	2	NA	1
Reshape	horizontal or vertical information flow?	2	NA	0.867
Dropout	visualize function, tensors, or both?	3	NA	1
Dropout	dropout function symbol shape?	4	2	0.28
Dropout	function symbol is 2d or 3d?	2	NA	1
Batchnorm	visualize function, tensors, or both?	3	NA	0.857
Batchnorm	function symbol shape?	3	2	0.373
Batchnorm	function symbol is 2d or 3d?	2	NA	1
Softmax	visualize function, tensors, or both?	3	NA	0.755
Softmax	function symbol shape?	4	2	0.265
Softmax	function symbol is 2d or 3d?	2	NA	1
Loss	loss function symbol shape?	3	2	0.355
Loss	function symbol is 2d or 3d?	2	NA	1
Update	update function symbol shape?	3	3	0.333
Update	function symbol is 2d or 3d?	2	NA	1

Chapter 6 The Design of VisRepNet

6.1 Design Process

We went through multiple rounds of symbol and language prototyping. During this process we constantly consulted with the perspectives of the user needs survey (Section 4), the user-defined symbolic design exercise (Section 5), the diagram design guidelines (Section 2.3), as well as our deep learning component research (Section 3).

Between each round of prototyping we also consulted with a professional designer who gave us tips. Generally this feedback centered around simplifying symbols and causing changes in color, line width, spacing, text size, etc.

6.2 VisRepNet Organization

In this section, we present the design of VisRepNet with the goal of representing deep neural networks effectively and intuitively. We describe VisRepNet's multiple-level architecture, and the symbol design for each component. We also present the reasons and insights behind the design of VisRepNet, taking in consideration the perspectives of the user needs survey (Section 4), the user-defined symbolic design (Section 5), the diagram design guidelines (Section 2.3), as well as our deep learning component research (Section 3).

VisRepNet is structured using procedural abstraction. So that if *function a* is within *function b*, then *function b* is at a higher level of procedural abstraction and is presented first. VisRepNet contains three procedural abstraction levels as follows.

- **Update Environment Level**. The first required level of abstraction is our top-down description of the network of interest and it's surrounding update environment. This describes the network as a black box and focuses on external information such as the input, output, loss function, optimization function, and update function. When a model includes multiple sequential methods of update then multiple update environment levels may be used.
- **Network Architecture Level**. The second required abstraction level begins to describe the network of interest by visualizing its specific make up of internal functions. When a model includes multiple networks of interest (Such as with GANs)

then multiple network architecture levels may be used.

- **Lower Function Level**. In the lower function level, we can separately describe functions that are within the network architecture. These can nest, as to describe functions within functions. Lower Function levels are not required, but are strongly suggested for complex networks.

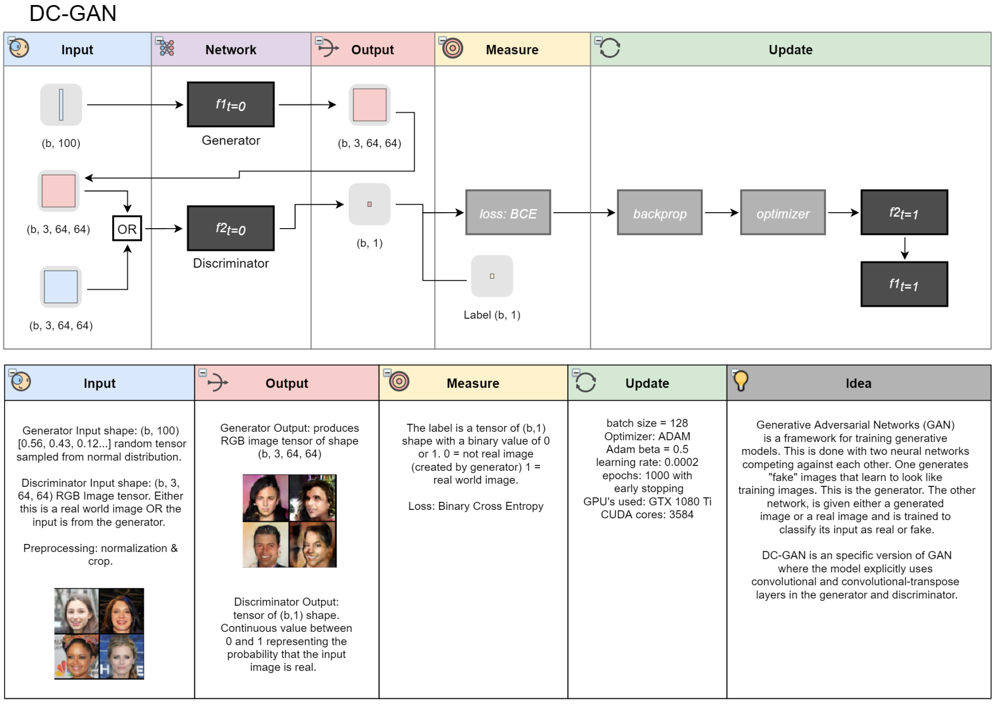
Usage of procedural abstraction is justified both by diagram design guidelines and user studies. User-needs required a lot of information to be shown, this information needed organization. The user needs survey indicated a high willingness for abstract groupings (95% choosing this option). Procedural abstraction also increases search ease and reduces cognitive load as everything is predictably grouped.

6.3 Update Environment Level

VisRepNet follows a very general framework for describing a deep neural network's update environment. This framework represents a linear process that we believe all deep neural networks share at a high abstract level. The framework goes as follows: the network receives *input*, *network* processes input, network produces *output*, output is *measured* based on some state, and then the network *updates* based on that measurement. We call this framework INOMU, which contains Input, Network, Output, Measure, Update stages. This INOMU framework then provides the update environment with an organizational skeleton, upon which details can be provided.

For each of INOMU's stages we created a small image that helps users conceptualize the framework within an understandable cognitive metaphor. This is done through an archery metaphor. Archery has a clear input (aiming/receiving visual input), a clear output (the arrow), and a clear measurement (the arrow is placed upon a target which indicates the accuracy). This made archery a good candidate as a conceptual metaphor. We then used a simple representation of a neural layer for network. And for update, two circling arrows are used, as this is common in user interface design and was commonly drawn by our users (50% drew similar types of circling arrow/s to indicate update.)

We use the update environment of DC-GAN in Figure 6.1 as an example to explain the update environment level in detail. The update environment level contains two sections. The upper section represents the update environment via a system diagram with functions and mathematical objects. The functions are simply represented by boxes with identifying names. The mathematical objects are represented with our visualization of



Input	Output	Measure	Update	Idea
Generator Input shape: (b, 100) [0.56, 0.43, 0.12...] random tensor sampled from normal distribution. Discriminator Input shape: (b, 3, 64, 64) RGB Image tensor. Either this is a real world image OR the input is from the generator. Preprocessing: normalization & crop.	Generator Output: produces RGB image tensor of shape (b, 3, 64, 64) Discriminator Output: tensor of (b,1) shape. Continuous value between 0 and 1 representing the probability that the input image is real.	The label is a tensor of (b,1) shape with a binary value of 0 or 1. 0 = not real image (created by generator) 1 = real world image. Loss: Binary Cross Entropy	batch size = 128 Optimizer: ADAM Adam beta = 0.5 learning rate: 0.0002 epochs: 1000 with early stopping GPU's used: GTX 1080 Ti CUDA cores: 3584	Generative Adversarial Networks (GAN) is a framework for training generative models. This is done with two neural networks competing against each other. One generates "fake" images that learn to look like training images. This is the generator. The other network, is given either a generated image or a real image and is trained to classify its input as real or fake. DC-GAN is an specific version of GAN where the model explicitly uses convolutional and convolutional-transpose layers in the generator and discriminator.

Figure 6.1 Example on the update environment of DC-GAN[98]

tensors (see section 6.4.1). In DC-GAN, both the generator and the discriminator functions are displayed in the Network block as they are both individually neural networks. The loss function and the label tensor are included in the Measurement block. The backpropagation function and optimization function belong to the Update block. The input and output tensors are placed in the Input and Output blocks respectively. These functions and mathematical objects together create a good system diagram of the networks update environment. As a bonus this is the same order that these functions are generally coded in.

The lower section of the update environment level consists of a series of textual blocks with the order of input, output, measure, update, and idea. In each of these blocks, the diagram author is expected to provide more details or explanations. The input block expects necessary descriptions of the input that include the tensor's shape, a demonstrative example, pre-processing methods, etc. The output block contains information about the output that includes the output tensor's shape, a demonstrative example, etc. The measure block provides the labels shape, a description of the label's origin, and a description of the loss function used for measurement. The update block contains information pertaining to the update mechanism, the optimizer, and (optionally) bench marking etc. Finally, the idea block summarizes the introduction, motivation, and contribution of the described deep neural network.

We created a textual area for VisRepNet for the following reasons. 70% of users wanted an input example visualized, 65% wanted to know about pre-processing functions. 75% wanted to have an output example visualized. And another 75% wanted to be able to see the loss equation. Results like these revealed that our users wanted a lot of information. An extra textural section, organized into input, output, measure, update, and idea blocks, provides an effective and flexible area to provide this information without cluttering our Update Environment system diagram.

Our creation of the idea partition was based on open feedback from users regarding academic papers. For example, P1 said *”If a novel recipe is being proposed, from the visualization diagram it should be immediately clear to the reader what made it different from the existing works.”* This idea partition is to provide an area where such information can be placed. Thus the author of an academic paper should use this area to outline the focus of his/her paper.

6.4 Components within the Network Architecture and Lower Function Levels

VisRepNet’s grammar for describing the architecture and functions of deep learning networks is simple and intuitive. VisRepNet describes the deep neural network in linear form: x –> f –> y. Compared to functional notation (y = f(x)), we believe that linear form reduces search difficulty and cognitive load. Further, we combine the linear form with the common left-to-right reading pattern (75% participants chose this option in the user need study).

Different from prior work[1-2] that utilized interconnected symbols to represent the connections between neurons, VisRepNet adopts discrete symbols to describe networks. That is to say, In VisRepNet, each symbol is a discrete symbol that relates to a specific function or mathematical object. For example, a linear layer is represented by a single symbol without any lines connecting it to the next layer. Discrete symbols offer many advantages. They are easier to isolate and recognize, which decreases cognitive load. They allow for a larger library of symbols and offer more flexibility when the need arises to add symbols. Finally, discrete symbols necessitate discrete concepts, which allows for easier learning and communication.

As discussed, procedural abstraction structures this visualization by having complex functions described in detail in separate sections below the Network Architecture Section. When these functions are referred to at higher levels, we use VisRepNet's general *function* symbol with a simple identification symbol within. The general *function* symbol is simply a box with an arrow running underneath it. The identification symbol is either a Greek letter or an Emoji symbol, depending on the authors choice.

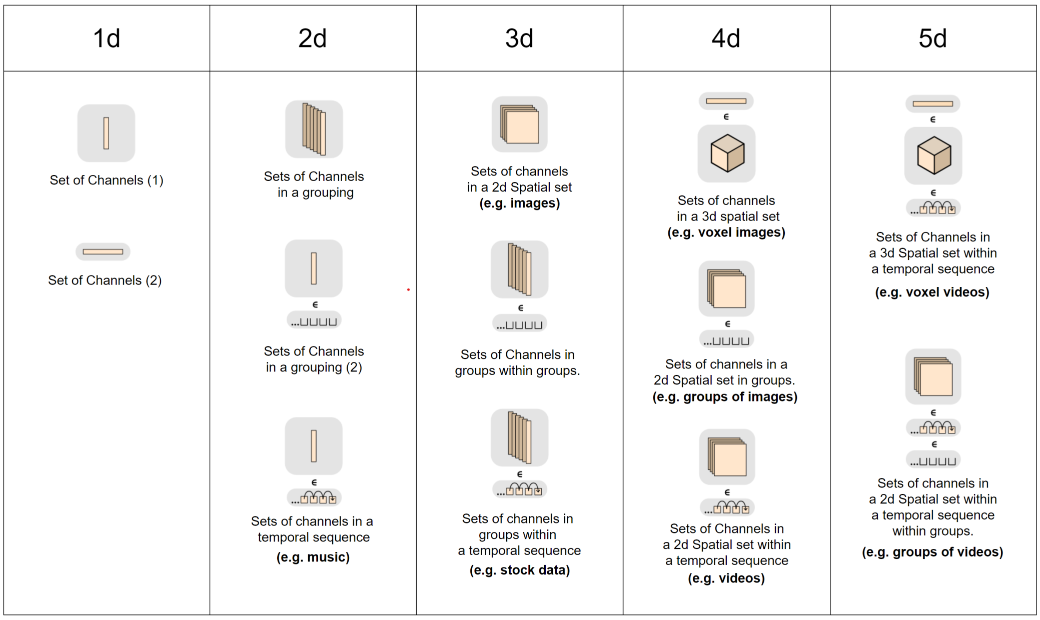

1d
2d
3d
4d
5d
Set of Channels (1)
Set of Channels (2)
Sets of Channels in a grouping
∈
...⊔⊔⊔⊔
Sets of Channels in a grouping (2)
∈
Sets of channels in a temporal sequence
(e.g. music)
Sets of channels in a 2d Spatial set
(e.g. images)
∈
Sets of Channels in groups within groups.
∈
Sets of channels in groups within a temporal sequence
(e.g. stock data)
∈
Sets of channels in a 3d spatial set
(e.g. voxel images)
∈
Sets of channels in a 2d Spatial set in groups.
(e.g. groups of images)
∈
Sets of Channels in a 2d Spatial set within a temporal sequence
(e.g. videos)
∈
∈
Sets of Channels in a 3d Spatial set within a temporal sequence
(e.g. voxel videos)
∈
∈
Sets of channels in a 2d Spatial set within a temporal sequence within groups.
(e.g. groups of videos)

Figure 6.2 This shows the combinatorial possibilities of VisRepNet's tensor representation system.

6.4.1 Tensors

In VisRepNet, simple tensors are visualized via stacks of rectangles of varying sizes. All tensors are colored, are described with colored text, and have a grey background behind them as to differentiate the group. The colors of tensors relate to INOMU. Blue is input, Red is output, and Orange is anything in between.

More complex tensors are represented via a combinatorial system as to represent niche cases as Figure 6.2 shows. This combinatorial system describes the real world interpretations that are placed upon tensors. For example, consider a tensor derived from a traffic camera network. Such a tensor can be described as a set of channels (RGB) within a 2d spatial set within a temporal sequence within a group set. Our visualization of tensors uses set theory symbols to combine representations of space, groups, and time as to describe such niche cases. The traffic-camera-network tensor could then be described by a stacked rectangle indicating x, y, and channel (We now have an image tensor). Then underneath, a *within* set theory symbol is followed by temporal sequence representation thus indicating time (We now have a video tensor). Underneath this, we can then have another *within* set theory symbol and then a grouping representation (We now have a group of videos). These more complex tensor representations allow for quick recognition of more complex tensor types.

Note, displaying tensors in between every single function creates unreasonably large diagrams and so we suggest representing tensors only when they change shape, or are followed by a convolutional layer. This allows users to follow the changing shape of tensors as they flow through the network without over complicating the diagram.

This representation of tensors was designed given our prior research. The user needs survey study and the user-defined design study, provided us with the following design recommendations for tensors.

1. We should visualize tensors shape (70% agreement in the user needs survey).
2. 2d tensors should be flat rectangles (80% agreement in the user-defined design study).
3. 3d tensors should be stacks of flat rectangles (53% agreement in the user-defined design study).
4. We do not need to visualize individual units (60% agreement in the user-defined design study).

The diagram design guidelines indicated that the tensors symbols should make use of spatial metaphors, be grouped visually and remain as simple as possible. Our theoretical generalizations of deep neural networks suggested a possible method to represent all possible tensor shapes, including niche ones. Thus in pursuit of completeness and the wish to represent all tensors, we developed a combinatorial system that makes use of set theory. This provides an option when representing more complex and niche tensors. Overall our tensor design satisfies all these various recommendations.

6.4.2 Convolutions

VisRepNet describes convolutions in their most general form as to apply our visualization more universally. For example, in many programming languages a ”fully connected” layer applied to a 3d tensor, actually has the ”fully connected” layer convolve over the third dimension[103]. Our convolutional representation allows us to explicitly visualize this distinction. This is achieved by representing the input and output kernel separately, and then further visualizing that kernel’s spatial, grouping, or temporal dimensions sep-

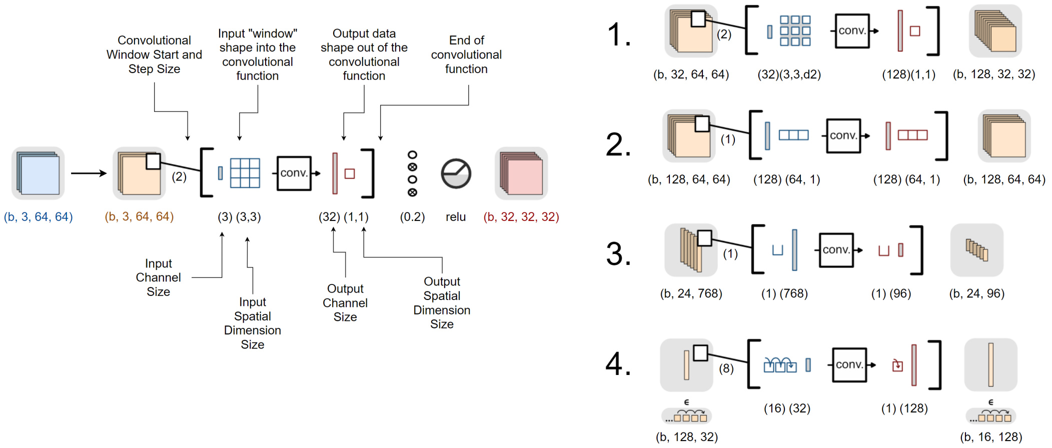


Figure 6.3 VisRepNet convolution examples: 1. Dilated convolution. 2. Convolution found in axial attention.[101] 3. Group based convolution. 4. Temporal convolution

arately. In this way we can visualize temporal convolutions, group based convolutions, and spatial convolutions as Figure 6.3 shows. The design of these kernels visualizations are similar to our tensor representations, except that they are flattened, and have no grey background. We color the kernels blue to indicate *input kernel*, and red to indicate *output kernel*.

These kernels are then framed by brackets indicating the beginning and end of the convolutional representation, with the beginning bracket overlaying the preceding tensor with a ’window’ representation. In between the kernels we visualize the convolutional function itself with a simple symbol which is labeled ”conv”. This symbol will be described in more detail in the following section.

This representation of convolutions was designed given our prior research. The user needs survey study and the user-defined design study, provided us with the following design recommendations for convolutions.

1. We should include information on the step size(60%), the kernel shape(75%), and the input and output dimensions (80%). (Agreement in the user needs survey)
2. We should represent the window/kernel on both the input and output side. (73% Agreement in user-defined design study)
3. The kernels symbols should be 2d (87% Agreement in user-defined design study)
4. We should have some ”arrow” connecting the kernels and indicating the direction of change. (60% Agreement in user-defined design study)

The diagram design guidelines then suggested that our convolution design should utilize the sliding window metaphor, be grouped visually and remain as simple as possible.

Meanwhile our Deep Learning research indicated: 1) we needed to represent the channel dimension within our convolution design; 2) there are a set of niche non-spatial convolutions that should also be represented. Thus in the pursuit of completeness, we wanted a method of representing channel, spatial, temporal and grouping based convolutions. From these various influences, we developed our convolution symbol system, which meets all of the above requirements.

6.4.3 Neural Layers

Neural layers have a box with an arrow running behind it. A label within the box describes the neural layer type. *Lin* for linear/fully connected layers, *Conv* for convolutional layers, *RNN* for recurrent layers e.t.c. VisRepNet is designed so that the representations of the mathematical objects around the neural layers provide the context that allows for quick recognition and understanding.

This representation of neural layers was designed given our prior research. The user needs survey study and the user-defined design study, provided us with the following design recommendations for neural layers.

1. Provide information on the normalization function (80%), the activation function (75%), and the input and output dimensionality of the tensors(75%). (Agreement in the user needs survey)
2. Tensor objects should be represented on both sides of the function. (73% Agreement in user-defined design study)
3. A rhombus shape and/or arrow should be used. (With the rhombus indicating channel change and the arrow indicating direction of causation) (60% Agreement in user-defined design study)
4. The symbol for the function should be 2d (93% Agreement in user-defined design study)

The diagram design guidelines suggested that our symbols be grouped visually and remain as simple as possible. Our end design conforms with all of the above specifications excepting one. We do not use a rhombus shape to indicate the compression or expansion of channels. Our earlier prototype designs did make use of this idea. We had a complex system of symbols that indicated expansion or compression in channel, space, groupings, and time. However this resulted in too much complexity, and is no longer necessary, as the surrounding tensors and kernels now provide this information.

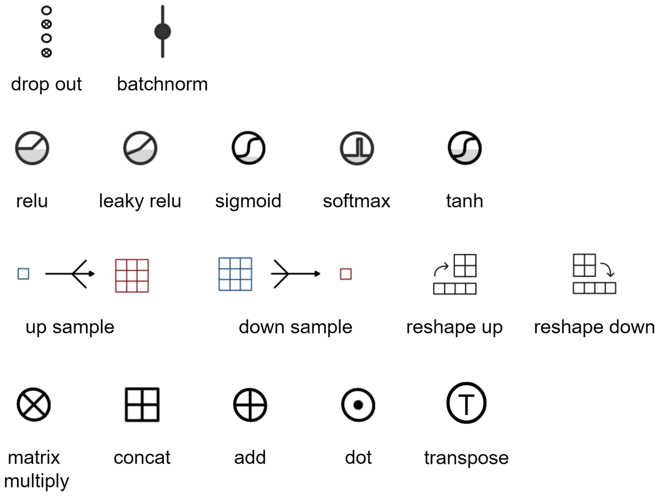


Figure 6.4 Symbols in VisRepNet for variety of functions.

6.4.4 Symbols for other functions

Figure 6.4 shows symbols for a variety of functions that are defined as follows:

1. Batch norm is represented by a line with a circle in the middle. This promotes the conceptual metaphor of values squishing towards zero.
2. Drop out is represented by four circles, with two crossed out. This represents certain neurons not being used.
3. Activation Functions are all encompassed by a circle and have a shaded region indicating values that are less then zero. Specific Activation functions are then represented via their shape on a graph.
4. Soft max function is excepted from the above rule. Instead of being represented by its ”S” shaped curve, it is represented by it’s effect on tensors. That is, one number is shown to have drastically increased, whilst all others have been pushed to zero.
5. Upsample and Downsample require a kernel on both sides of its function, just like convolutions. The actual function is represented via a trident symbol expanding or compressing the kernels.
6. Reshape requires tensors to be represented on either side of it. The reshape up symbol is visualized via a rectangle turning into a square. The reshape down symbol is visualized via a square being turned into a rectangle.
7. The various linear algebra functions keep their common representational form.

Given these representations, the next few paragraphs will provide our reasoning for each these symbol’s designs.

Our representation of regularization functions was designed given our prior research.

The user needs survey study and the user-defined design study, provided us with the following design recommendations for regularization functions.

1. Provide information on regularization functions (80% Agreement in the user needs survey)
2. Symbol should be 2d (100% Agreement in user-defined design study)
3. Symbol does not need tensors on either side (100% Agreement in user-defined design study)
4. Dropout symbol shows crossed out or blacked out neurons (40% Agreement in user-defined design study) (not enough categorical agreement for requirement)
5. Batch norm symbol shows box with label (46% Agreement in user-defined design study) (not enough categorical agreement for requirement)
6. Batch norm symbol indicating the squashing of values (38% Agreement in user-defined design study) (not enough categorical agreement for requirement)

The diagram design guidelines suggested we use the two conceptual metaphors associated with batchnorm and dropout. With batchnorm, there is a mathematical metaphor, where a grouping takes place in the middle of a number line. With drop out, there is conceptual metaphor of nodes being crossed out and not being used. Our end design meets all of the above suggestions except using a box with a label to describe batchnorm. We do not do this, as we already use this symbol to describe non-specific functions that require labelling.

VisRepNet's representation of upsample and downsample was designed given our prior research. The user needs survey study and the user-defined design study, provided us with the following design recommendations for upsample and downsample.

1. Up and Down sampling are requested to be visualized. (75%) (Agreement in the user needs survey)
2. Tensor objects are represented on both sides of the function.(86% Agreement in user-defined design study)
3. Function symbol is some type of arrow connecting the tensors (71% Agreement in user-defined design study)
4. Function symbol is 2d in design (100% Agreement in user-defined design study)

Using these user derived ideas and the diagram design guidelines we created a trident/arrow symbol to symbolize up and down sample. With the required kernels on either side of the up and down sample symbol, the trident/arrow indicates a compression or expansion. Despite many of our users drawing a simple arrow, We did not do this as such a symbol would be too similar to others and create confusion.

Our representation of reshape up and down was designed given our prior research.The user needs survey study and the user-defined design study, provided us with the following design recommendations for reshape.

1. Reshape is requested to be visualized. (70%) (Agreement in the user needs survey)
2. Tensor objects are represented on both sides of the function.(86% Agreement in user-defined design study)
3. Function symbol is some type of arrow connecting the tensors (67% Agreement in user-defined design study)
4. Function symbol is 2d in design (100% Agreement in user-defined design study)

The user-defined symbols for reshape and up/down sample were similar, and we thus have similar recommendations. Once again, we did not use the user suggested arrow symbol, as this is too common and would cause confusion. The diagram design guidelines suggest using the *reshape* conceptual metaphor, where one shape is turned into another. Thus we use a simple symbol of a square turning into a rectangle. Due to user-defined data we also require the visualization of the input and output tensors to be on either side of our reshape symbol.

In regard to activation functions, we only collected user-defined design data on the Softmax activation function. This is because the diagram design guidelines suggest using symbols that are already well known within the community. Most activation functions have signs resembling their graphed function that are well established. Softmax however has a ”s” shape that is too similar to Sigmoid and Tanh to use. Given this, the user needs survey study and the user-defined design study, provides us with the following design recommendations for Activation Functions.

1. Activation functions are requested to be visualized. (75% Agreement in the user needs survey)
2. Softmax is represented via a box with a label. (36% Agreement in user-defined design study) (not enough categorical agreement for requirement)
3. Softmax is represented via ”S” curve. (29% Agreement in user-defined design study) (not enough categorical agreement for requirement)
4. Softmax is represented via an indication of Softmax’s effect.(21% Agreement in user-defined design study) (not enough categorical agreement for requirement)

5. Tensor objects are NOT represented with the Softmax function (86% Agreement in user-defined design study)
6. Function symbol is 2d in design (100% Agreement in user-defined design study)

The first two suggested user symbols are already in use within VisRepNet. Thus we follow the third most common user option and represent softmax via its effect (enhance one value, decrease all others). Given the diagram design guidelines, we also group all activation functions via a circle border, and differentiate Sigmoid and Tanh via our shading of values below zero.

VisRepNet's representations of Linear Algebra functions was designed given our prior research. The user needs survey provides us with the following design recommendations for Linear Algebra Functions.

1. Visualize the dot product. (85% Agreement in the user needs survey)
2. Visualize adding two tensors together. (85% Agreement in the user needs survey)
3. Visualize concatenating a tensor. (85% Agreement in the user needs survey)
4. Visualize slicing a tensor.(80% Agreement in the user needs survey)
5. Visualize matrix multiplication of two tensors. (75% Agreement in the user needs survey)

We did not collect user-defined design data on Linear Algebra Functions. This is because the diagram design guidelines require us to use well known symbols if available. The above referents all have well known symbols within the community.

6.4.5 Undefined Representation or Symbol

We acknowledge that VisRepNet does not provide a symbol for every function related to deep neural networks. When there is a missing symbol for a desired function, we suggest the following methods of representation. If the function is similar to a function with a symbol, consider using that symbol with a textual label below. If using an abbreviated label, please create a key section underneath all other sections, where one would define the abbreviation. If no similar function exists, please use a label with our general ”function” symbol of a box with an arrow running underneath it.

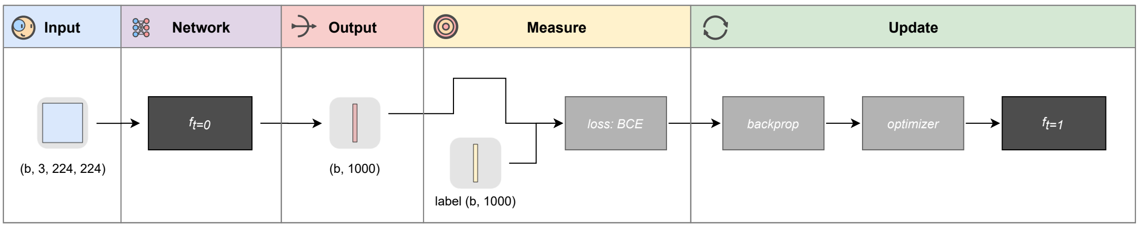
6.5 Examples

We provide four examples of VisRepNet. These include:

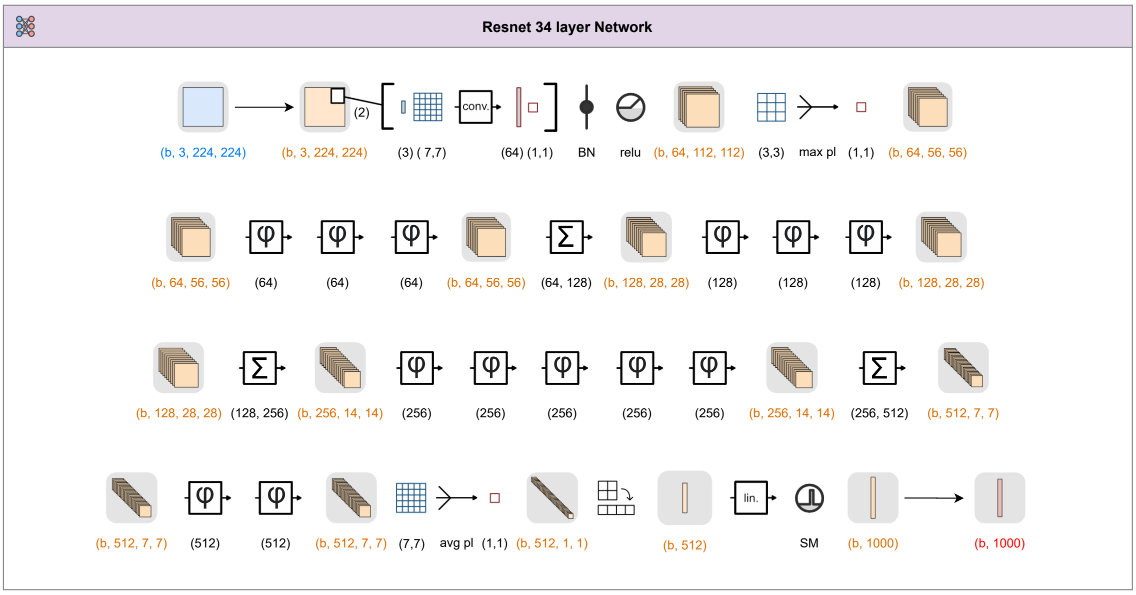
1. **Resnet 34 layers**[95], This diagram shows how a deep architecture can be displayed

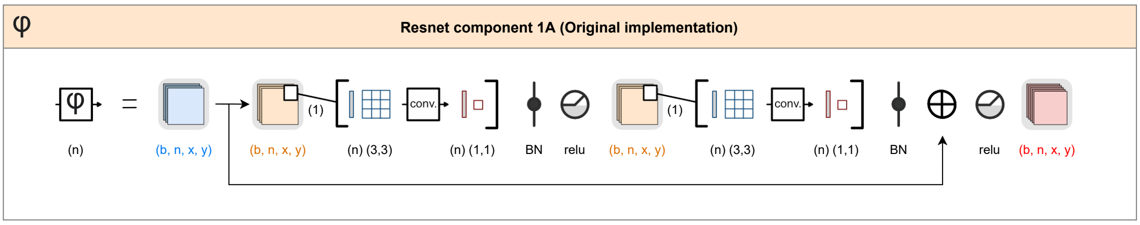
with procedural abstraction. 6.5

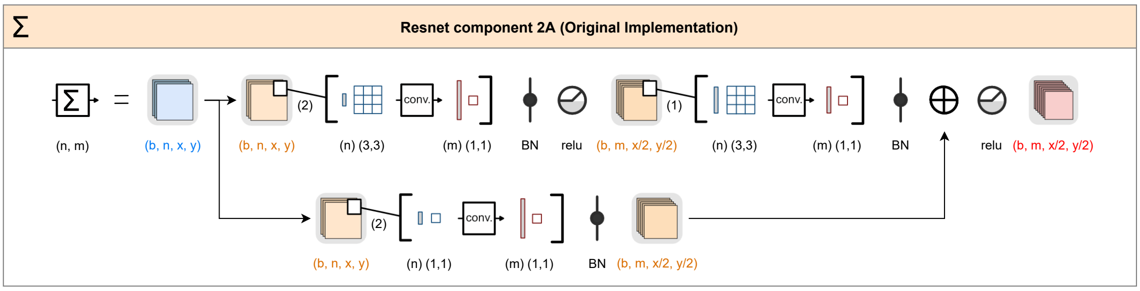
2. **DC-GAN**[98], This diagram shows the value in displaying the learning environment through the INOMU framework. We can visualize and explain the adversarial nature of GAN training in our VisRepNet diagram. 6.5
3. **BERT**[100], This diagram shows the subtle design decisions that went into the BERT architecture. With the VisRepNet method, readers can see and consider each component within complex language models. 6.5
4. **SimCLR**[99], this diagram shows again how the INOMU framework can be useful in explaining how complex training environments work. It also demonstrates how different training stages can be visualized within the VisRepNet framework. 6.5

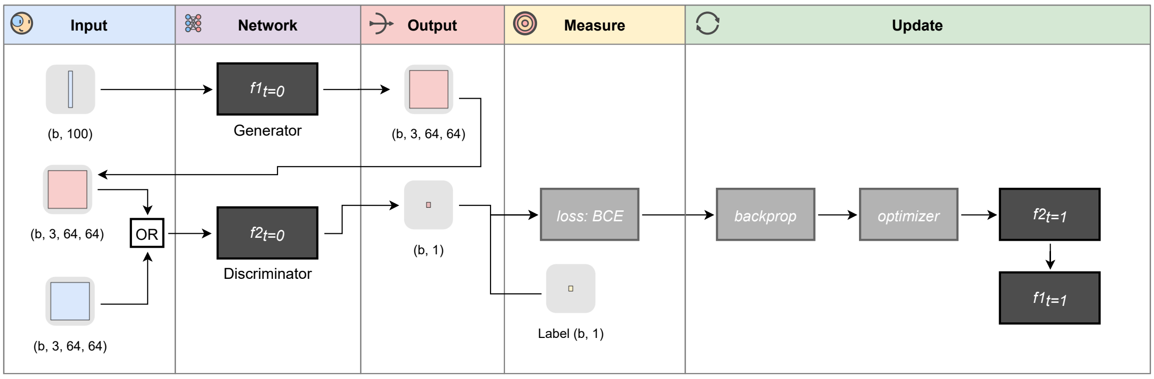


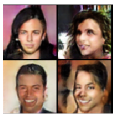
Input	Output	Measure	Update	Idea
Input is a (b, 3, 224, 224) RGB image tensor Prepossessing: normalization, random crop, random rotation color jiter. Data set: ILSVRC 2012	Output is a (b, numOfClasses) continuous tensor. e.g. [0.1, 0.01, 0.99, 0.05 ...] Each value represents the probability that the class type is in the picture.	Label is a (b, numOfClasses) continuous tensor e.g. [0, 0, 1, 0 ...] It is a Human curated label for each picture indicating which classes are present.. Loss: Binary Cross Entropy Results: Top-1 error: 26.73 Top-5 error: 8.74	batch size = 16 Optimizer: ADAM learning rate: 0.001 epochs: 1000 with early stopping GPU's used: GTX 1080 Ti CUDA cores: 3584	Resnet is a categorical encoder. It takes in an image input and determines the probability that a class/category is present in that input image. The main innovation of resnet is its skip layers. The skip layers do two things. First, they allow the gradient to more easily propagate through the network allowing for larger networks. Two, The skip layers act upon the input tensor in a new way. They allow for specialized iterative changes. That is each skip block can specialize to act upon a certain part of the tensor, and then additively change the "working draft" that is supplied by the skip layers.

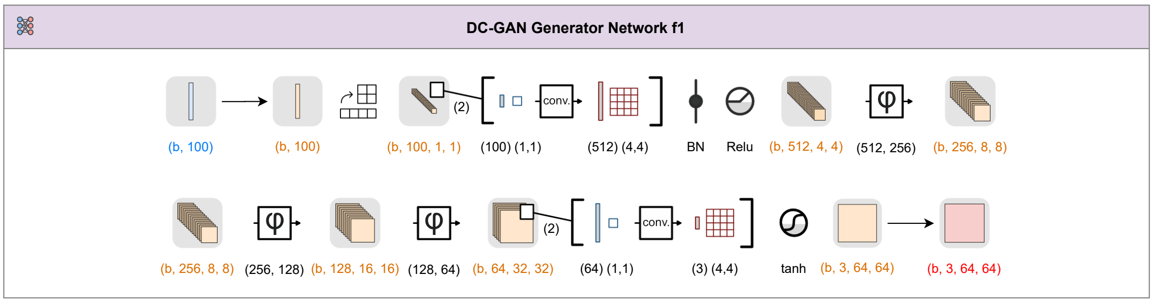


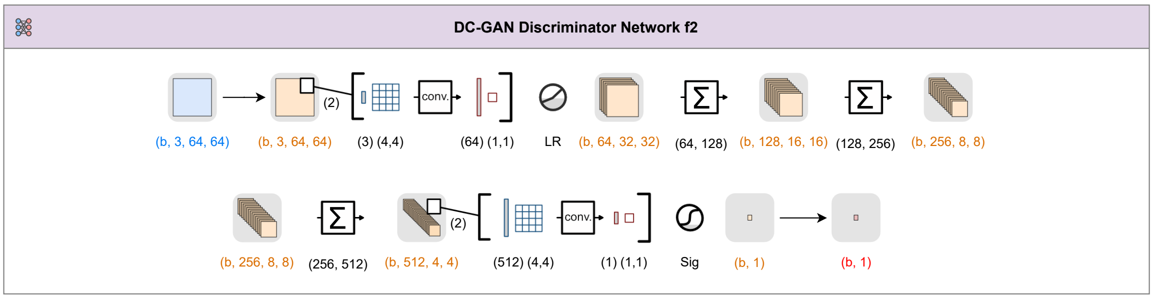


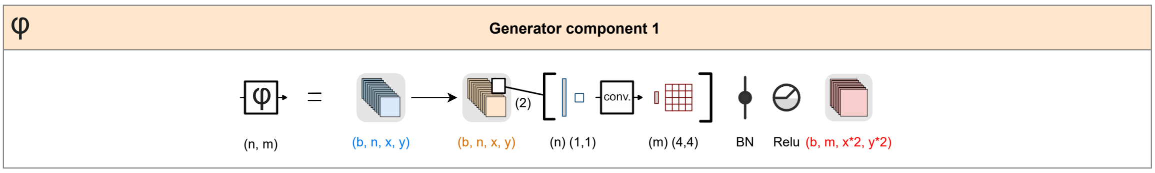


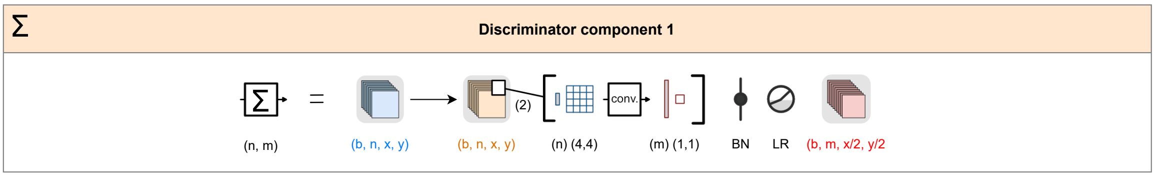


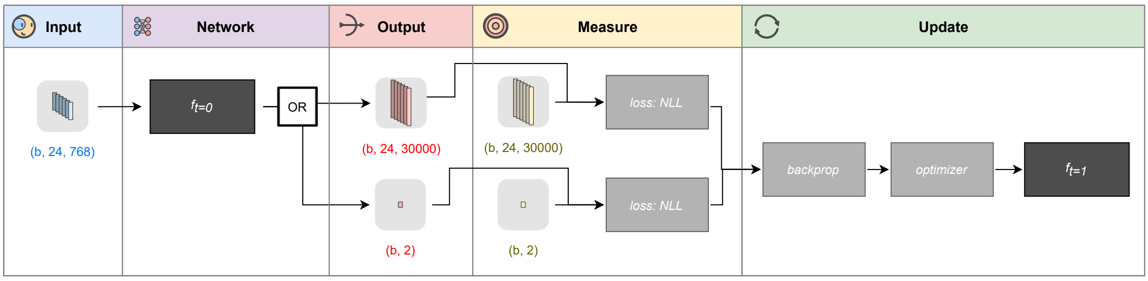
Input	Output	Measure	Update	Idea
Generator Input shape: (b, 100) [0.56, 0.43, 0.12...] random tensor sampled from normal distribution. Discriminator Input shape: (b, 3, 64, 64) RGB Image tensor. Either this is a real world image OR the input is from the generator. Preprocessing: normalization & crop. 	Generator Output: produces RGB image tensor of shape (b, 3, 64, 64)  Discriminator Output: tensor of (b,1) shape. Continuous value between 0 and 1 representing the probability that the input image is real.	The label is a tensor of (b,1) shape with a binary value of 0 or 1. 0 = not real image (created by generator) 1 = real world image. Loss: Binary Cross Entropy	batch size = 128 Optimizer: ADAM Adam beta = 0.5 learning rate: 0.0002 epochs: 1000 with early stopping GPU's used: GTX 1080 Ti CUDA cores: 3584	Generative Adversarial Networks (GAN) is a framework for training generative models. This is done with two neural networks competing against each other. One generates "fake" images that learn to look like training images. This is the generator. The other network, is given either a generated image or a real image and is trained to classify its input as real or fake. DC-GAN is an specific version of GAN where the model explicitly uses convolutional and convolutional-transpose layers in the generator and discriminator.



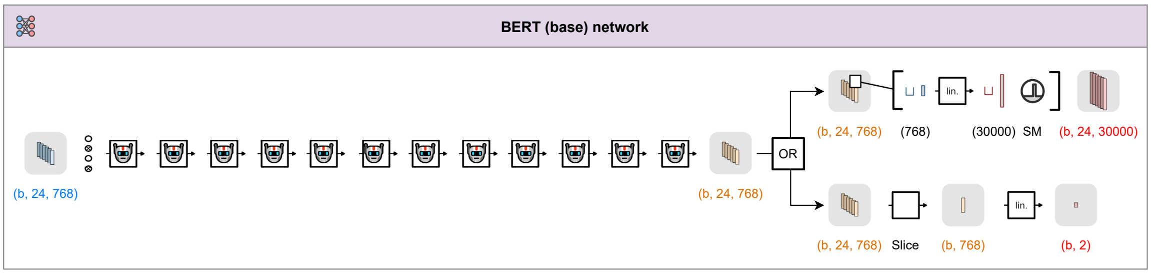


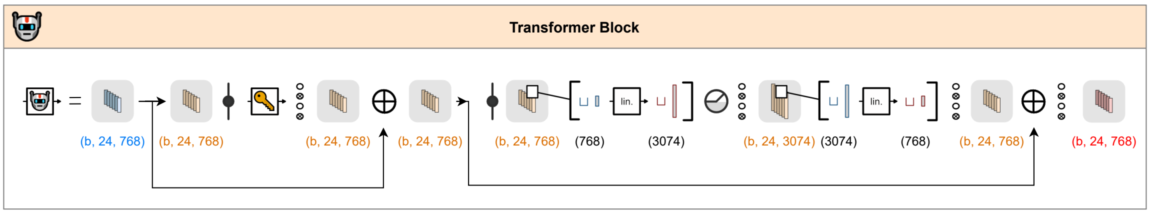


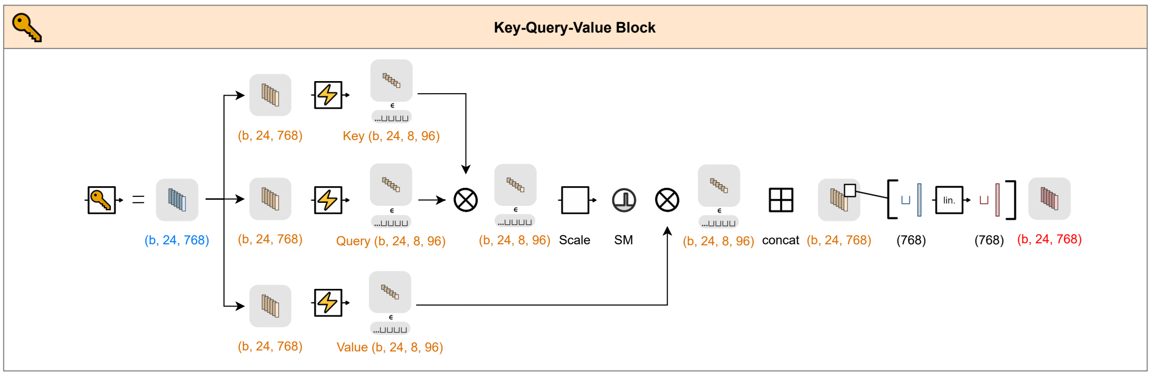


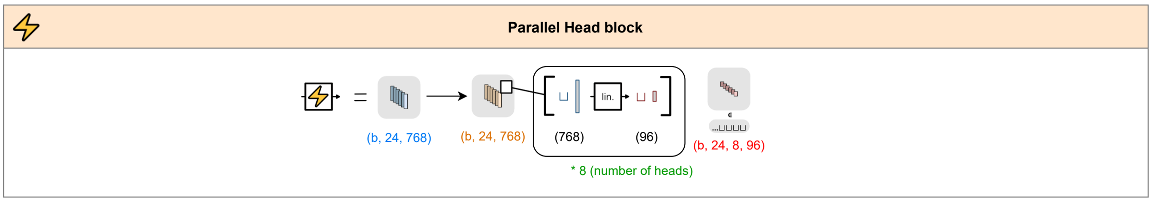


Input	Output	Measure	Update	Idea
Input is a (b, 24, 768) tensor representing a sentence with 24 tokens represented in 768 dimensional space. To do this. First, we tokenize each word. With a sentence start and end token (plus padding) resulting in 24 tokens. Second, We project each token into 768 dimensional space by combining: • A token embedding • A positional embedding • A segment embedding	Two possible outputs for different training modes. 1. (b, 24, 30000) tensor this output is a sentence embedding like the input. (24 tokens in 768 dimensional space) 2. (b, 2) tensor, this yes / no output allows BERT to be trained as a classifier.	Two possible labels for different training modes. 1. (b, 24, 30000) output tensor is measured against the input sentence (fill in the gaps training), or in translation models the label is the input sentence in a different language. 2. (b, 2) tensor is compared to human curated binary labels (ex. Spam vs. Not Spam) Loss = Negative Log Likelihood	Batch size = 8 Optimizer: ADAM learning rate: 0.0001 with 0.01 weight decay. GPU's used: TITAN RTX Results: GLUE score: 73.5 SQuAD, test EM: 85.1 SQuAD, test F1: 91.8	BERT stands for Bidirectional Encoder Representations from Transformers. Unlike RNN's, LSTMs, or GRUs, BERT was one of the first NLP models to take the entire sentence as input in one go, thus making its connections "bidirectional" BERT also makes good use of transformers. Transformers have a clever Key, Query, Value mechanism that acts as an attention layer, focusing the models attention on specific representations at one time. This is combined with skip layers so that the model iteratively attends to and modifies the input tensor as it changes into the final output tensor.



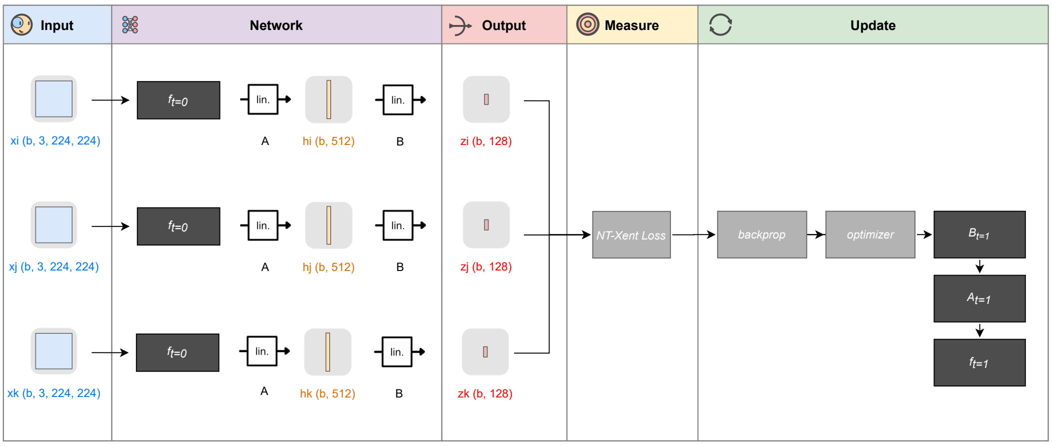




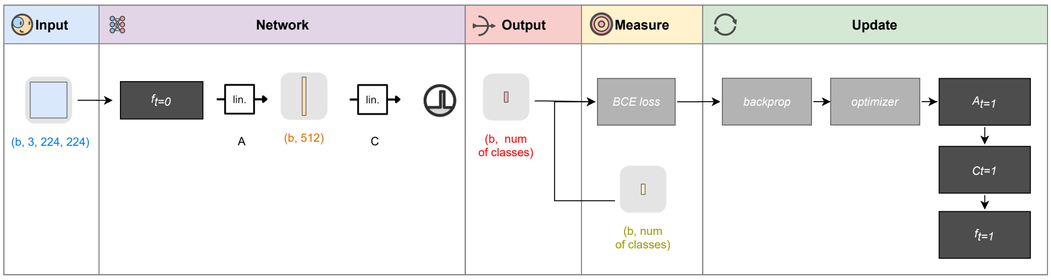


SimCLR (With transfer learning)

Stage 1 - Self-Supervised Contrastive Training



Stage 2 - Supervised fine-tuning



Input	Output	Measure	Update	Idea
Stage 1: Three Inputs of shape (b, 3, 224, 224). Two of these are derived from the same image but have been altered/augmented in different ways. (random crop, color distortion, Gaussian blur). The third image is derived from a different image and is treated as a negative example. (actually from the same batch as the others) **Stage 2:** input is of shape (b, 3, 224, 224) and is derived from an image.	**Stage 1:** Three outputs are created (in parallel) these are latent space embeddings of the three inputs respectively. z_i = latent embedding from image 1 - A z_j = latent embedding from image 1 - B z_k = latent embedding from image 2 - A **Stage 2:** Output is a tensor of (b, num of classes) shape. The output represents the networks certainty that the input is apart of x class.	**Stage 1:** The NT_Xent loss pushes the latent spaces derived from the same image to be close together. Meanwhile it pushes the latent spaces that are derived from different images further apart. $\mathcal{L}(\mathbf{z_i}, \mathbf{z_j}) = -\log \frac{\exp(\mathbf{z_i}\mathbf{z_j}/\tau)}{\sum_{k=1}^{2N} 1_{k \neq i} \exp(\mathbf{z_i}\mathbf{z_k}/\tau)}$ **Stage 2:** Label is human curated and presents ground truth of which class an image belongs to. The output is compared to the label with BCE loss.	Base network = ResNet-50 batch size = 4096 Optimizer: ADAM Epochs = 1000 Results: CIFAR10: 97.7 CIFAR100 85.9 Caltech-101: 93.3	SimCLR is a contrastive learning framework which can be used with any base architecture. Contrastive learning pushes latent representations from the same image to be similar, whilst seperating latent representations of different images. This enables self-supervised training. Some findings from the SimCLR paper are: 1. Contrastive learning improves with greater image augmentation 2. Contrastive Learning benefits from larger batch sizes. 3. A temperature parameter in the loss function reduces the need to negative-sample mine.

Base network
Can use any base network. (ResNet-50 is used for benchmarking)

Chapter 7 Evaluation

We evaluated VisRepNet's performance through a between-subject user study. This section describes the experiment design, procedure, participants and results. Overall VisRepNet achieved statistically significant results in all areas tested when compared to the control.

7.1 Experiment Design

We designed this experiment to evaluate VisRepNet's performance regarding its ability to help with learning and user perceived performance. We utilized an A - B test, with the test method being VisRepNet, and a control method using a set of available resources on the Internet. We used this hodgepodge of internet resources to make the comparison fair, since VisRepNet presents more information than other alternatives[1-2,5-29]. The control consisted of:

1. Any diagrams that were provided by the academic paper that introduced the network.[95,98]
2. A popular online diagrammatic form representing the network.[104]
3. An online summary about the network.[105-106]
4. A PyTorch[103] print out of the network.

Together these resources provide a comparable amount of information as VisRepNet. If not equal, the control provides more information. We argue this hodgepodge control works because these are likely the types of resources that one would find when searching on the internet. Thus this is a natural alternative to VisRepNet.

It is worth mentioning that each user was introduced to the VisRepNet diagram for the first time during testing. We did not teach them anything about VisRepNet. Thus the meaning of VisRepNet's grammar, symbols, symbol systems and procedural abstraction levels was entirely new to them. Our reasoning for this is two fold: 1) We wanted to test the intuitiveness of VisRepNet. 2) We wanted to avoid the biases due to teaching.

In this experiment, we chose the original 34-layer Resnet network, and the original DC-GAN network as our test contents. These two networks were chosen for the following reasons. We had a diverse user group with varying experience levels. These two networks offer some challenging aspects whilst remaining somewhat accessible to users with little experience. Resnet's challenging aspects relate to its architecture, whilst DC-Gan's challenging aspects relate to its update environment. Thus, by testing both these networks we hoped to keep our test accessible and varied.

We measured the ability to help with learning by counting the incremental knowledge gained when given access to one specific visualization method. We first obtained a baseline score via a multiple-choice test with three questions for each of the two deep neural networks. These tests were done without any materials. Then each participant learned Resnet and DC-GAN using VisRepNet and the control method separately. Finally, we tested the participant's learning using the same multiple-choice test for the two deep neural networks. Thus, we have an A - B test, where each visualization method is tested on its ability to improve the participant's score from their baseline.

The multiple-choice test had following questions:

1. Resnet is famous for its skip connections. Sometimes this skip connection has to be spatially compressed. How does the 34 layer original Resnet achieve this? *A. Max pool B. Average pool C. A (1,1) convolution with a stride of 2 D. A (3,3) convolution with a stride of 2 E. I do not know*
2. A general rule of thumb in the deep learning community says that the activation function should happen before batch normalization. Does the 34 layer original Resnet follow this convention? *A. Yes B. No C. Sometimes D. I do not know*
3. What is the final layer and activation function of the 34 layer original Resnet? *A. A convolutional layer with sigmoid activation B. A convolutional layer with softmax activation C. A convolutional layer with relu activation D. A linear layer with sigmoid activation E. A linear layer with softmax activation F. A linear layer with relu activation G. I do not know*
4. What is the end activation function of the Generator network in the original DC-GAN network? *A. tanh B. Sigmoid C. Relu D. Leaky Relu E. Softmax F. I do not know*
5. What is the best interpretation of the output of the discriminator network in the original DC-GAN network? *A. The output is an image tensor of (b, 3, 64, 64) shape B. The output is a continuous tensor of (b, 1) shape indicating the model's belief that the input was a real image vs. a generated image. C. the output is a binary tensor of (b, 1) shape indicating the model's belief that the input was a real image vs. a generated image. D. the output is a tensor of (b, 2) shape indicating the models*

belief that the input was a real image vs. a generated image E. I do not know

6. What is the best interpretation of the input of the generator network in the original DC-GAN network? *A. the input is a real world image tensor of (b, 3, 64, 64) shape B. the input is a generated image tensor of (b, 3, 64, 64) shape C. the input is a real world OR generated image tensor of (b, 3, 64, 64) shape D the input is a (b, 100) latent vector derived from an image embedding E. the input is a (b, 100) random vector sampled from a normal distribution F. the input is a (b, 100) random vector sampled from a binomial distribution G. I do not know*

To measure the user perceived performance and preference, we briefly explained VisRepNet after the multiple choice test. Then we showed them another example of a network represented by both VisRepNet and the control method. Finally, participants were asked to rate each of the methods with the following questions using a 7-point Likert scale (1: strongly disagree, 4: neutral, 7: strongly agree).

1. (**Specificity**) I can quickly and easily find the specific information I need about the deep neural network.
2. (**Completeness**) The material provided enough information about the deep neural network.
3. (**Easy-to-understand**) I found the deep neural network easy to understand via the material.
4. (**Usefulness-for-coding**) I would find the material useful if I had to code the deep neural network.
5. (**Usefulness-for-learning**) I would find the material valuable if I had to learn about the deep neural network for the first time.

We should also note that we included two attention check questions within the evaluation for quality control.

7.2 Participants

We recruited 18 participants (3 females, 15 males) with an average age of 25.5 (s.d. = 2.5). 8 participants were graduate students, 2 participants were undergraduate students and 8 participants were software engineers. Participants had an average of 2.4 (s.d. = 1.1) years of learning experience in deep learning. Their self-reported deep learning knowledge has an average score of 4.0 (s.d. = 1.2) using 7-point Likert scale.

7.3 Experiment Procedure

We first introduced each participant to the purpose and procedure of the experiment. Each participant was asked to fill in a questionnaire about their demographic information and experience in deep learning. This was followed by the baseline multiple-choice tests for both Resnet and DC-GAN. Next each participant self-learned one of the two deep neural networks with either VisRepNet or with the control method. The two visualization methods were counter-balanced among the participants. Participants were required to learn only with the provided materials. After each participant finished their self-learning, he/she then filled in the same multiple-choice test again.

Participants were then given a brief explanation about VisRepNet and were given the user preference questions. Users were allowed to revisit and modify the user preference survey. Finally, we asked participants about their subjective feedback about VisRepNet. The whole experiment took around 40 minutes. Each participant was given a 20 USD gift card for their time and effort.

7.4 Result

We threw out one user's data, due to the failure of the attention check. For the ability of VisRepNet to help with learning, we utilized Mann-Whitney U Test for between-subject significance analysis (p < 0.05). For the user preference evaluation, we utilized Wilcoxon Signed-Rank Test for within-subject significance analysis (p < 0.05).

7.4.1 Ability to help with learning.

Figure 7.1A shows the score difference to the baseline using the different visualization methods. Results show that VisRepNet outperforms the control visualization method significantly ($U = 80.5, Z = -2.28, p = 0.02$).

7.4.2 User perceived performance.

Figure 7.1B shows the user perceived performance between VisRepNet and the control method along five different metrics. Results show that VisRepNet ($avg. = 6.0, s.d. = 1.0$) outperforms the control method ($avg. = 4.1, s.d. = 1.4$) when users want to quickly and easily find specific information about the deep neural network (**Specificity**, $Z = -3.3, p < 0.001$). VisRepNet ($avg. = 6.1, s.d. = 0.6$) outperforms the control method ($avg. = 5.2, s.d. = 1.2$) regarding providing enough information about

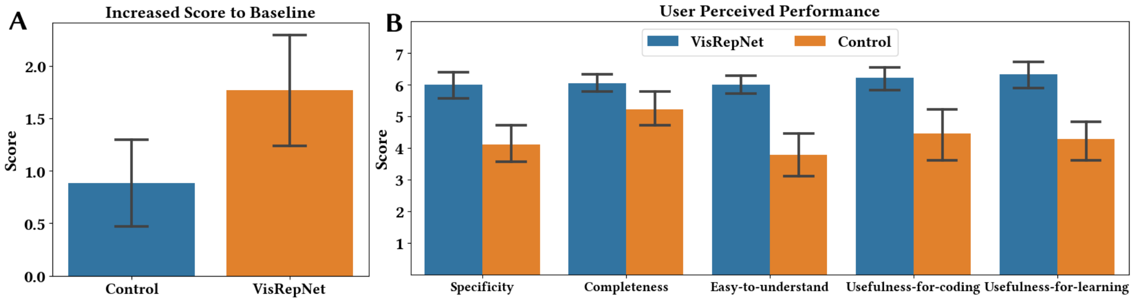


Figure 7.1 A: Measuring ability to help with learning using increased score from baseline. B: Five user preference metrics between the two visualization methods.

the deep neural network (**Completeness**, $Z = -2.2, p = 0.02$). Users found VisRepNet ($avg. = 6.0, s.d. = 0.7$) to be more easy to understand (**Easy-to-understand**, $Z = -3.5, p < 0.001$) compared to the control method ($avg. = 3.8, s.d. = 1.5$). Users consider VisRepNet ($avg. = 6.2, s.d. = 0.7$) to be more useful when implementing the deep neural network (**Usefulness-to-coding**, $Z = -2.9, p = 0.004$) than the control method ($avg. = 4.4, s.d. = 1.8$). Further, we found that users find VisRepNet ($avg. = 6.3, s.d. = 0.9$) to be more helpful for learning (**Usefulness-to-learning**, $Z = -3.4, p < 0.001$) when compared to the control method ($avg. = 4.3, s.d. = 1.4$).

Furthermore, we collected user's subjective feedback on VisRepNet regarding our system of representing tensors, our system of representing convolutions, the organization of sections, and whether we should use emojis or Greek symbols to identify functions.

Regarding our system of representing tensors P1 commented ”[The] User probably needs time to get used to these symbols”, P2 commented ”It took me awhile to understand all the symbols”. This was expected given that we did not teach our participants. On the other hand P3 commented ”They are really clear and straightforward especially for the representation of TIME. In general, the temporal sequence is hard to be described or represented in diagrams.”

Regarding our system of representing convolutions we had similar comments. P4 said ”The expanded format for convolutions takes getting used to.” P5 said ”its pretty complicated without instruction.” Multiple users commented upon the window, the bracket, and the step size representations as a source of confusion. In future innovations, we would consider changing this. P4 suggested representing the window “with a distance measurement symbol like on technical diagrams rather than a square.”

Regarding our organization of sections (e.g. procedural abstraction) P2 commented ”Overall I think it is clear. I like the order those sections are represented. It helps me see the big picture first and shows the details later.” whilst another user P6 said ”its difficult to read when matching the sub-diagram to the main diagram."

The question of whether to use Emoji's or Greek letters to represent functions was a divisive one. P7 said "I prefer 100x more emojis to indicate complicated blocs explained somewhere else than Greek letters." On the other hand, P2 commented "No… I prefer text index." 54% of those who commented on this, preferred emojis. Thus, it was a rather equal split.

Chapter 8 Discussion and Future Work

VisRepNet aims to provide an organised framework via which users can quickly understand deep neural networks. VisRepNet visualizes the training environment and the internal architecture of neural networks at both a high and low level, whilst also providing a place for authors to discuss their focus. VisRepNet is the first visualization method, that we know of, to have been designed via user centered design methodology whilst specializing towards the communication and learning of complex neural networks. Overall, our feedback was positive and our tests provided significant results.

Our multiple choice experiment showed that VisRepNet was rather intuitive for our users. Without having seen the language before, our users' scores improved greater when given the VisRepNet diagram when compared to the control diagram. This was validating, however our users demographic is important to consider here. For instance we would not expect deep learning beginners to intuitively understand our language. Indeed by our estimation, the value of VisRepNet increases with the complexity of the network. For beginners there are simpler visualizations for simpler networks. VisRepNet shines when visualizing neural networks that approach or exceed the complexity of BERT. With current methods, such networks are often not visualized in detail. Such an undertaking generally results in a visualizations too chaotic for easy analysis. With VisRepNet however, the complexity of BERT and networks like it, become more accessible. For example, without diving into the code, it would be very difficult to find out that BERT's attention mechanism has 8 heads which operate in parallel. In our VisRepNet diagram, one can find this detail rather quickly.

We acknowledge that VisRepNet can not represent all deep neural networks. The field is growing too fast and in too many directions to ever be sure of that. We do however suggest that our methods go a long way towards approaching completeness. At each opportunity VisRepNet goes to great lengths to visualize the general form of concepts. This is exemplified by INOMU, our convolutional representation system, and our tensor representation system. We also promote and encourage the visualization of unrepresented functions, via the use of current symbols with new labels. Finally we encourage flexibility and growth via our setting up of a GitHub page for community driven improvement.

Having said all of this, there are two potential completeness issues that we have found with the current system. First, we predict problems in the representation of tensors within graph networks. We did not spend much time exploring this problem. Perhaps future work could expand upon our tensor representation system to include graph tensors. The second problem relates to hyper-parameters. There are some hyper-parameters that VisRepNet does not, so far, represent. For example, the padding size of convolutional layers. These hyper-parameters were not labeled as important enough for representation within our user studies, however a complete language to describe neural networks should include them.

For VisRepNet we see three clear pathways towards improvement. First, we would revisit the larger organization of VisRepNet's placement onto the page. The textual section is too squashed, and multiple users complained about the small text size. Perhaps we can expand these out, by allowing the diagram to go onto multiple pages. Second, we would like to improve the user experience of diagram authors. We are creating a set of instructions, templates and symbol libraries on GitHub towards this end. These templates and libraries are to be used in the open-source program draw.io[107]. It is by far the best option that we have found for creating VisRepNet diagrams. Third, we would like to set up a method for community driven improvement of VisRepNet. On our VisRepNet GitHub page we intend to provide a set of guidelines which allow for forking, and then eventual merging. Inspired by INOMU, we would like VisRepNet to update over time given measurable improvements.

We hope this trend of improving scientific representations picks up in other areas of science as well. While science is the abstraction of real-world phenomena into models that can be tested, there has always been another practice of taking those testable models and presenting them in a manner for learning and communication. The design of diagrams, of language, and of how to present a subject, are important in their own right. Currently we, as a scientific community, do not give this the attention it deserves. Too many students are learning about complex subjects as presented by those who initially delved into those complex subjects. This is not optimal, as the people who explore the frontier of science are often not the best people to explain it (excepting Feynman). If we wish to continually improve our methods of representation, we can. Design Thinking and User Center Design now provides a set of techniques and methods that are testable and reliable, and we suggest their use in this endeavor.

Chapter 9 Conclusion

In this paper, we presented Visually Representing Networks (VisRepNet), a diagrammatic visualization for describing deep neural networks which can help users learn, understand, and communicate neural network architectures and their training environments. We begun by categorizing components from a varied set of complex, neural network, architectures. We then created diagram design guidelines from a compilation of research into diagram design, semiotics, and cognitive linguistics. We next obtained user's needs and dimensions of interest in regard to deep neural network visualizations. This user survey study indicated that users want visualizations that include information about networks' architecture, update environment, functions and hyper-parameters. Further, we extracted design recommendations from a user-defined symbol drawing study. Fully considering user' needs and design guidelines, we then designed an organizational structure of procedural abstraction levels. Further, based on the diagram design guidelines and user-defined design recommendations, we designed symbols that represent components in the deep neural network. Given this work, VisRepNet not only provides a high-level view of the network architecture and training environment, but also an intuitive low-level detailed view of functions and hyperparameters. Finally, we evaluated VisRepNet's effectiveness through a quantitative and qualitative user study. Results show that VisRepNet significantly outperforms current deep neural network visualization methods regarding user preference and its ability to help with learning.

Bibliography

[1] LeNail A. Nn-svg: Publication-ready neural network architecture schematics[J/OL]. Journal of Open Source Software, 2019, 4: 747. DOI: 10.21105/joss.00747.

[2] Iqbal H. Harisiqbal88/plotneuralnet v1.0.0[CP/OL]. Zenodo, 2018. https://doi.org/10.5281/zenodo.2526396.

[3] Casner S, Larkin J. Cognitive efficiency considerations for good graphic design[J]. 1989: 12.

[4] Amar R A, Stasko J T. Knowledge precepts for design and evaluation of information visualizations[J/OL]. IEEE Transactions on Visualization and Computer Graphics, 2005, 11(4): 432-442. DOI: 10.1109/TVCG.2005.63.

[5] Bauerle A, Ropinski T. Net2vis: Transforming deep convolutional networks into publication-ready visualizations[J]. arXiv preprint arXiv:1902.04394, 2019.

[6] Gavrikov P. visualkeras[J/OL]. GitHub. https://github.com/paulgavrikov/visualkeras/.

[7] Isaksson M. dotnets[J/OL]. GitHub. https://github.com/martisak/dotnets.

[8] Graph visualization software[J/OL]. Graphviz. https://graphviz.org/.

[9] Michel J, Holbrook Z, Grosser S, et al. [J/OL]. ENNUI Elegant Neural Network User Interface . https://math.mit.edu/ennui/.

[10] Fyles M. What does machine learning look like?[J/OL]. Graphcore, 2017. https://www.graphcore.ai/posts/what-does-machine-learning-look-like.

[11] Liu Y, Zhu C, Botime, et al. [J/OL]. TensorSpace.js. https://tensorspace.org/.

[12] David Gschwend S D. Netscope cnn analyzer[EB/OL]. https://dgschwend.github.io/netscope/quickstart.html.

[13] Lajtoš M. Moniel - interactive notation for computational graphs[EB/OL]. https://mlajtos.github.io/moniel/.

[14] Karim R. Illustrated: 10 cnn architectures[J/OL]. Medium, 2020. https://towardsdatascience.com/illustrated-10-cnn-architectures-95d78ace614d.

[15] Migdal P. Simple diagrams of convoluted neural networks[J/OL]. Medium, 2019. https://medium.com/inbrowserai/simple-diagrams-of-convoluted-neural-networks-39c097d2925b.

[16] Harris D. Prototypeml - visual neural network design platform for pytorch[EB/OL]. https://www.prototypeml.com/.

[17] Ding G W. drawconvnet[J/OL]. GitHub. https://github.com/gwding/draw_convnet.

[18] Tensorboard[J/OL]. TensorFlow. https://www.tensorflow.org/tensorboard.

[19] Jia Y, Shelhamer E, Donahue J, et al. Caffe: Convolutional architecture for fast feature embedding[J]. arXiv preprint arXiv:1408.5093, 2014.

[20] MATLAB. version 7.10.0 (r2010a)[M]. Natick, Massachusetts: The MathWorks Inc., 2010.

[21] Chen L. keras-js[J/OL]. GitHub. https://github.com/transcranial/keras-js.

[22] Migdał P. keras-sequential-ascii[J/OL]. GitHub. https://github.com/stared/keras-sequential-ascii/.

[23] Roeder L. Netron[J/OL]. GitHub. https://github.com/lutzroeder/Netron.

[24] Team K. Keras documentation: Model plotting utilities[J/OL]. Keras. https://keras.io/api/utils/model_plotting_utils/.

[25] Conx[J/OL]. Conx documentation. https://conx.readthedocs.io/en/latest/index.html.

[26] Wagenaar T. [J/OL]. Neataptic.js - Home. https://wagenaartje.github.io/neataptic/.

[27] Fauske K M. Neural network texample[J/OL]. TeXample.net. https://texample.net/tikz/examples/neural-network/.

[28] Bian J. Quiver[M/OL]. GitHub, 2016. https://github.com/keplr-io/quiver.

[29] Wongsuphasawat K, Smilkov D, Wexler J, et al. Visualizing dataflow graphs of deep learning models in tensorflow[J/OL]. IEEE Transactions on Visualization and Computer Graphics, 2018, 24(1): 1-12. DOI: 10.1109/TVCG.2017.2744878.

[30] Alber M, Lapuschkin S, Seegerer P, et al. innvestigate neural networks![Z]. 2018.

[31] Harley A W. An interactive node-link visualization of convolutional neural networks[C]// Bebis G, Boyle R, Parvin B, et al. Advances in Visual Computing. Cham: Springer International Publishing, 2015: 867-877.

[32] Krause J, Perer A, Ng K. Interacting with predictions: Visual inspection of black-box machine learning models[M/OL]. New York, NY, USA: Association for Computing Machinery, 2016: 5686–5697. https://doi.org/10.1145/2858036.2858529.

[33] Ledesma S, Ibarra-Manzano M, Garcia-Hernandez M, et al. Neural lab a simulator for artificial neural networks[C/OL]// 2017: 716-721. DOI: 10.1109/SAI.2017.8252175.

[34] Rauber P, Fadel S, Falcão A, et al. Visualizing the hidden activity of artificial neural networks[J/OL]. IEEE Transactions on Visualization and Computer Graphics, 2016, 23: 1-1. DOI: 10.1109/TVCG.2016.2598838.

[35] Tzeng F Y, Ma k l. Opening the black box - data driven visualization of neural networks[C/OL]// 2005: 383 - 390. DOI: 10.1109/VISUAL.2005.1532820.

[36] Seifert C, Aamir A, Balagopalan A, et al. Visualizations of deep neural networks in computer vision: A survey[M/OL]. Cham: Springer International Publishing, 2017: 123-144. https://doi.org/10.1007/978-3-319-54024-5_6.

[37] Zhang Q s, Zhu S C. Visual interpretability for deep learning: a survey[J]. Frontiers of Information Technology & Electronic Engineering, 2018, 19(1): 27-39.

[38] Shahroudnejad A. A survey on understanding, visualizations, and explanation of deep neural networks[J]. arXiv preprint arXiv:2102.01792, 2021.

[39] Nguyen A, Yosinski J, Clune J. Understanding neural networks via feature visualization: A survey[M]// Explainable AI: interpreting, explaining and visualizing deep learning. Springer, 2019: 55-76.

[40] Smilkov D, Thorat N, Nicholson C, et al. Embedding projector: Interactive visualization and interpretation of embeddings[J]. 2016.

[41] Lukas Biewald S L, Chris Van Pelt. Weights and biases[EB/OL]. https://wandb.ai/site.

[42] Kahng M, Andrews P Y, Kalro A, et al. Activis: Visual exploration of industry-scale deep neural network models[Z]. 2017.

[43] Cashman D, Patterson G, Mosca A, et al. Rnnbow: Visualizing learning via backpropagation gradients in rnns[J]. IEEE Computer Graphics and Applications, 2018, 38(6): 39-50.

[44] Selvaraju R R, Cogswell M, Das A, et al. Grad-cam: Visual explanations from deep networks via gradient-based localization[C]// Proceedings of the IEEE international conference on computer vision. 2017: 618-626.

[45] Ancona M, Ceolini E, Öztireli C, et al. Towards better understanding of gradient-based attribution methods for deep neural networks[J]. arXiv preprint arXiv:1711.06104, 2017.

[46] Mahendran A, Vedaldi A. Visualizing deep convolutional neural networks using natural pre-images[J/OL]. International Journal of Computer Vision, 2016, 120. DOI: 10.1007/s11263-016-0911-8.

[47] Simonyan K, Vedaldi A, Zisserman A. Deep inside convolutional networks: Visualising image classification models and saliency maps[Z]. 2014.

[48] Cao C, Liu X, Yang Y, et al. Look and think twice: Capturing top-down visual attention with feedback convolutional neural networks[C]// Proceedings of the IEEE International Conference on Computer Vision (ICCV). 2015.

[49] Dosovitskiy A, Brox T. Inverting visual representations with convolutional networks[C]// Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2016.

[50] Erhan D, Courville A, Bengio Y. Understanding representations learned in deep architectures[J]. Department dInformatique et Recherche Operationnelle, University of Montreal, QC, Canada, Tech. Rep, 2010, 1355(1).

[51] Grün F, Rupprecht C, Navab N, et al. A taxonomy and library for visualizing learned features in convolutional neural networks[J]. arXiv preprint arXiv:1606.07757, 2016.

[52] Nguyen A, Yosinski J, Clune J. Multifaceted feature visualization: Uncovering the different types of features learned by each neuron in deep neural networks[J]. arXiv preprint arXiv:1602.03616, 2016.

[53] Simonyan K, Vedaldi A, Zisserman A. Deep inside convolutional networks: Visualising image classification models and saliency maps[J]. arXiv preprint arXiv:1312.6034, 2013.

[54] van der Maaten L, Hinton G. Visualizing high-dimensional data using t-sne. 4[J]. 2008.

[55] Sainburg T, Thielk M, Gentner T Q. Latent space visualization, characterization, and generation of diverse vocal communication signals[J]. bioRxiv, 2019: 870311.

[56] Liu S, Maljovec D, Wang B, et al. Visualizing high-dimensional data: Advances in the past decade[J]. IEEE transactions on visualization and computer graphics, 2016, 23(3): 1249-1268.

[57] Kahng M, Thorat N, Chau D H, et al. Gan lab: Understanding complex deep generative models using interactive visual experimentation[J]. IEEE transactions on visualization and computer graphics, 2018, 25(1): 310-320.

[58] Jäckle D, Hund M, Behrisch M, et al. Pattern trails: visual analysis of pattern transitions in subspaces[C]// 2017 IEEE Conference on Visual Analytics Science and Technology (VAST). IEEE, 2017: 1-12.

[59] Xia J, Ye F, Zhou F, et al. Visual identification and extraction of intrinsic axes in high-dimensional data[J]. IEEE Access, 2019, 7: 79565-79578.

[60] Kammer D, Keck M, Gründer T, et al. Glyphboard: Visual exploration of high-dimensional data combining glyphs with dimensionality reduction[J]. IEEE transactions on visualization and computer graphics, 2020, 26(4): 1661-1671.

[61] Carter S, Smilkov D. Tensorflow - neural network playground[J/OL]. A Neural Network Playground. https://playground.tensorflow.org/.

[62] Strobelt H, Gehrmann S, Huber B, et al. Visual analysis of hidden state dynamics in recurrent neural networks[J/OL]. CoRR, 2016, abs/1606.07461. http://arxiv.org/abs/1606.07461.

[63] Karpathy A, Johnson J, Fei-Fei L. Visualizing and understanding recurrent networks[Z]. 2015.

[64] Ákos Kádár, Chrupała G, Alishahi A. Representation of linguistic form and function in recurrent neural networks[Z]. 2016.

[65] Li J, Chen X, Hovy E, et al. Visualizing and understanding neural models in nlp[Z]. 2016.

[66] Vig J. A multiscale visualization of attention in the transformer model[J]. arXiv preprint arXiv:1906.05714, 2019.

[67] Chefer H, Gur S, Wolf L. Transformer interpretability beyond attention visualization[J]. arXiv preprint arXiv:2012.09838, 2020.

[68] van Aken B, Winter B, Löser A, et al. Visbert: Hidden-state visualizations for transformers[J]. arXiv preprint arXiv:2011.04507, 2020.

[69] Nadin M. Design and semiotics[J]. 1987.

[70] Sweller J. Cognitive load theory and e-learning[C]// Biswas G, Bull S, Kay J, et al. Artificial Intelligence in Education. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011: 5-6.

[71] Andrews E. Semiotic principles in cognitive neuroscience[M/OL]. 2019. DOI: 10.5772/intech open.89791.

[72] Ergonomics of human-system interaction: part 210 : human-centred design for interactive systems; 2nd ed.[M/OL]. Geneva: ISO, 2019. https://cds.cern.ch/record/2684407.

[73] Abras C, Maloney-Krichmar D, Preece J, et al. User-centered design[J]. Bainbridge, W. Encyclopedia of Human-Computer Interaction. Thousand Oaks: Sage Publications, 2004, 37(4): 445-456.

[74] Vredenburg K, Mao J Y, Smith P W, et al. A survey of user-centered design practice[C]// Proceedings of the SIGCHI conference on Human factors in computing systems. 2002: 471-478.

[75] Garrett J J. The elements of user experience: user-centered design for the web and beyond[M]. Pearson Education, 2010.

[76] Holtzblatt K, Wendell J B, Wood S. Rapid contextual design: a how-to guide to key techniques for user-centered design[M]. Elsevier, 2004.

[77] Feynman R P. Space-time approach to non-relativistic quantum mechanics[J/OL]. Rev. Mod. Phys., 1948, 20: 367-387. https://link.aps.org/doi/10.1103/RevModPhys.20.367.

[78] Staats K. iconji[EB/OL]. http://www.iconji.com/.

[79] Chan E, Seyed T, Stuerzlinger W, et al. User elicitation on single-hand microgestures[M/OL]. New York, NY, USA: Association for Computing Machinery, 2016: 3403–3414. https://doi.org/10.1145/2858036.2858589.

[80] Ferron M, Mana N, Mich O. Designing mid-air gesture interaction with mobile devices for older adults[M/OL]. Cham: Springer International Publishing, 2019: 81-100. https://doi.org/10.1007/978-3-030-06076-3_6.

[81] Heydekorn J, Frisch M, Dachselt R. Evaluating a user-elicited gesture set for interactive displays[C]// Mensch Computer. 2011.

[82] Kou Y, Kow Y M, Cheng K. Developing intuitive gestures for spatial interaction with large public displays[C]// HCI. 2015.

[83] Wobbrock J, Morris M, Wilson A D. User-defined gestures for surface computing[C]// CHI. 2009.

[84] Freitas E. The diagram as story: Unfolding the event-structure of the mathematical diagram[J]. 2012, 32: 27-33.

[85] Radford L, Schubring G, Seeger F. Signifying and meaning-making in mathematical thinking, teaching, and learning[J/OL]. Educational Studies in Mathematics, 2011, 77: 149-156. DOI: 10.1007/s10649-011-9322-5.

[86] Goguen J. An introduction to algebraic semiotics, with application to user interface design[C]// International Workshop on Computation for Metaphors, Analogy, and Agents. Springer, 1998: 242-291.

[87] Lakoff M, G. Johnson. Metaphors we live by.[C]// University of Chicago Press., 1981.

[88] Parsons P C. Conceptual metaphor theory as a foundation for communicative visualization design[C]// IEEE VIS Workshop on Visualization for Communication (VisComm 2018). 2018.

[89] Blackwell A F. Pictorial representation and metaphor in visual language design[J]. Journal of Visual Languages & Computing, 2001, 12(3): 223-252.

[90] Blackwell A F. The reification of metaphor as a design tool[J]. ACM Transactions on Computer-Human Interaction (TOCHI), 2006, 13(4): 490-530.

[91] Blackwell A. Metaphor in diagrams[C]// 1998.

[92] Parsons P. Conceptual metaphor theory as a foundation for communicative visualization design[C]// 2018.

[93] Kadunz G, Straber R. Image–metaphor–diagram: Visualisation in learning mathematics.[J]. International Group for the Psychology of Mathematics Education, 2004.

[94] Shah P. Graph comprehension: The role of format, content and individual differences[M/OL]. London: Springer London, 2002: 173-185. https://doi.org/10.1007/978-1-4471-0109-3_10.

[95] He K, Zhang X, Ren S, et al. Deep residual learning for image recognition[Z]. 2015.

[96] Szegedy C, Wei Liu, Yangqing Jia, et al. Going deeper with convolutions[C/OL]// 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2015: 1-9. https://ieeexplore.ieee.org/document/7298594. DOI: 10.1109/CVPR.2015.7298594.

[97] Ronneberger O, Fischer P, Brox T. U-net: Convolutional networks for biomedical image segmentation[C]// Navab N, Hornegger J, Wells W M, et al. Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015. Cham: Springer International Publishing, 2015: 234-241.

[98] Radford A, Metz L, Chintala S. Unsupervised representation learning with deep convolutional generative adversarial networks[Z]. 2016.

[99] Chen T, Kornblith S, Norouzi M, et al. A simple framework for contrastive learning of visual representations[Z]. 2020.

[100] Devlin J, Chang M W, Lee K, et al. Bert: Pre-training of deep bidirectional transformers for language understanding[Z]. 2019.

[101] Ho J, Kalchbrenner N, Weissenborn D, et al. Axial attention in multidimensional transformers[Z]. 2019.

[102] Wobbrock J, Aung H, Rothrock B, et al. Maximizing the guessability of symbolic input[C/OL]// 2005: 1869-1872. DOI: 10.1145/1056808.1057043.

[103] Paszke A, Gross S, Chintala S, et al. Automatic differentiation in pytorch[C]// NIPS-W. 2017.

[104] Karim R. Illustrated: 10 cnn architectures[J/OL]. Medium, 2020. https://towardsdatascience.com/illustrated-10-cnn-architectures-95d78ace614d#e4b1.

[105] Jordan J. Common architectures in convolutional neural networks.[J/OL]. Jeremy Jordan, 2018. https://www.jeremyjordan.me/convnet-architectures/#resnet.

[106] PyTorch. Dcgan tutorial[J/OL]. DCGAN Tutorial - PyTorch Tutorials 1.7.1 documentation. https://pytorch.org/tutorials/beginner/dcgan_faces_tutorial.html.

[107] Draw.io[J/OL]. Diagram Software and Flowchart Maker. https://www.diagrams.net/.

Appendix A Appendix

A.0.1 Multi-answer Questions for User Needs survey

1. You want the visualization of the network...
    - A. along the x axis with left = input side and right = output side
    - B. along the x axis with right = input side and left = output side
    - C. along the y axis with up = input side and down = output side
    - D. along the y axis with down = input side and up = output side
2. Do you want the visualization to move to new lines (like writing on a page) or do you want it to continue on one line perhaps requiring scrolling to see.
    - A. move to new lines
    - B. continuous on one line
3. Do you want common sets of layers and functions to be defined separately. With the main visualization referencing these common sets.
    - A. Yes
    - B. No
4. Neural networks transform an input tensor as it travels through the network. How would you like this shape information conveyed. Choose all that apply.
    - I want to see the shape of the tensor at different points in the network. ex (batch, 32, 256, 256)
    - I want to see the change in tensor shape from different functions in the network. ex. (b *1, capacity * 2, x /2, y/2)
    - I would like to see a representation of both. They are both useful.
    - I do not need to see the shape of the tensor at any point in the network.
    - other...
5. You are studying a new neural network architecture, Do you want to see the Input? if so, How do you want the input represented? Choose all that apply.
    - I dont really need/want the input represented
    - I want to see the general type of input
    - I want to see the tensor shape of the input
    - I want to see an example of the input before preprocessing.
    - I want to see an example of the input after preprocessing.

- If there are multiple inputs I want to see where in the architecture these enter.
- other...

6. Do you want to see what preprocessing effects the input before entering the model? Choose all that apply.
    - I dont really need to see the preprocessing
    - I want to see generally which preprocessing functions were implemented
    - I want to see which preprocessing functions were implemented and the exact hyper-parameters used.
    - other...
7. There is a Linear / Dense / Fully Connected, Neural Network Layer. You get a quick look at it. What do you want to see? Choose all that apply. Choose all that apply.
    - I want to see the Bias
    - I want to see the number of weights
    - I want to see if there is a general compression or expansion in dimensions/channels
    - I want to see the exact input and output number of dimensions/channels
    - I want to see the following Activation Function if there is one
    - I want to see the following Normalization Function if there is one
    - I want to see the code level initialization method, and any other such methods.
    - other...
8. There is a convolutional neural network layer. You get a quick look at it. What do you want to see? Choose all that apply. Choose all that apply.
    - I want to see the Bias
    - I want to see the number of weights
    - I want to see if there is a general compression or expansion in dimensions/channels
    - I want to see the exact input and output number of dimensions/channels
    - I want to see the following Activation Function if there is one
    - I want to see the following Normalization Function if there is one
    - I want to see the kernel size/shape of the convolution
    - I want to see the step size of the convolution
    - I want to see if the kernel is dilated.
    - I want to see the code level initialization method, and any other such methods.

- other...

9. You are studying a new neural network architecture, which of these dimensions of interest do you want to see if they are implemented within the model. Choose all that apply.
    - adding two tensors together
    - finding the dot product between two tensors
    - transposing a tensor
    - Matrix Multiplying two tensors.
    - Concatenating two tensors
    - Slicing a tensor
    - Reshaping a Tensor
    - Up Sampling
    - Down Sampling
    - All of the above
    - Other...
10. You are studying a new neural network architecture, Do you want the output represented, how do you want the Output represented? Choose all that apply.
    - I want to see the tensor shape of the output
    - I want to see the general type of the output.
    - I want to see an example of the output
    - I dont really need/want the output represented
    - If there are multiple outputs I want to see where in the network architecture they are produced.
    - other...
11. You are studying a new neural network architecture, Do you want the loss represented, how do you want the loss represented? Choose all that apply.
    - I want to see the general loss type
    - I want to see the loss's name
    - I want to see an equation for the loss.
    - I don't really need to see the loss
    - other...
12. You are studying a new neural network architecture, Do you want the label represented, how do you want the label represented? Choose all that apply.

- I want to see the tensor shape of the label
- I want to see the general type of the label.
- I want to see an example of the label
- I dont really need/want the label represented
- other...

13. Is there anything else that you wish to mention related to what you want out of a neural network diagram?

Acknowledgements

A.1 Acknowledgements

I sincerely thank my supervisor Professor Jie Tang and Associate Professor Yuntao Wang for their meticulous guidance. I have learned so much from their advice and counsel. I am sure the lessons learnt will benefit me throughout my life.

I would also like to thank the Global Innovation Exchange for providing this opportunity. The help and support of faculty, students, and personnel has been much appreciated.

声 明

本人郑重声明：所呈交的学位论文，是本人在导师指导下，独立进行研究工作所取得的成果。尽我所知，除文中已经注明引用的内容外，本学位论文的研究成果不包含任何他人享有著作权的内容。对本论文所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确方式标明。

I solemnly declare that the thesis submitted is the result of my independent research work under the guidance of my supervisor. To the best of my knowledge, the research results do not contain contents of any others who have claimed copyrights except for those cited in the thesis. Other individuals and organizations, who have contributed to the research work, have been clearly identified in the thesis.

签 名：________ 日 期：2020/03/30

Resume and Academic Achievements

Joshua Clancy was born on November the 15th, 1993 in Sydney, New South Wales, Australia.

He began his bachelor’s study in the Department of Economics, in the University of Washington in September 2014, majoring in economics, and receiving a Bachelor of Science degree in July 2017.

He began his dual degree master’s study in the Global Innovation Exchange, University of Washington in September 2018, and got a Master of Science degree in Technology and Innovation in July 2020.

This thesis paper is in support of his second masters degree from the Global Innovation Exchange at Tsinghua University. He has pursued this degree since September, 2018.

Appendix B COMMENTS FROM THESIS SUPERVISOR

This paper presents a visually representing networks (VisRepNet) to help users understand deep neural network. It collected a compilation of research into diagram design, semiotics and cognitive linguistics to create diagram design guidelines. It conducts user studies to investigate user's needs and design guidelines. VisRepNet can not only represent high-level overview of the network's architecture and update environment, but also low-level detailed view of its function and hyperparameters.

Overall, the paper is well-written and the content is comprehensive.

Appendix C　答辩委员会决议书

论文提出了……

Joshua Clancy’s thesis explores the visualization of neural networks with his project: VisRepNet. This project combines user research with theoretical findings in deep learning and diagram design in order to design the VisRepNet visualization. VisRepNet is then evaluated quantitatively and qualitatively.

论文取得的主要创新性成果包括：

1. This research investigated user’s needs and dimensions of interest for deep neural network visualizations through a user survey study.

2. This research created VisRepNet, a diagrammatic visualization to describe deep neural networks. VisRepNet design is guided and justified by user needs, user-defined symbolic designs, deep learning research and diagram design guidelines.

3. This research evaluated VisRepNet’s effectiveness through quantitative and qualitative user studies. Results show that VisRepNet significantly outperforms current deep neural network visualization methods regarding its ability to help with learning and user perceived performance.

The oral defense was fluently presented and the questions were clearly answered. Through anonymous voting, the defense committee consisting of 5 members agrees that the oral defense is successfully passed, and suggests that the degree of Master of Science in Engineering be awarded to Joshua Clancy.